제2권 통권4호 1989년3월1일 발행(매월 1일 발행) 1988년10월10일 등록(등록번호 라-3794) ㊥140-150 서울특별시 용산구 갈월동 17

하이테크 퍼스널컴퓨터

89/03



특집 /

MS-DOS의 구조

LX-800의 버그

배치화일로 인스톨작업을

키보드의 스캔코드 값을 구한다

확장 함수키의 정의

유틸리티 소개/**아카이브**

회로해석이 가능한 만능 캐드

마이크로 캡(Ⅱ)

누구나가 쉽게 배울 수 있는 초보자 페이지

디렉토리 구조

컴퓨터 그는 누구인가

배치화일

□ **편집자노트** □

각급학교 컴퓨터교육 신중히 검토되어야

한 국가의 행정이 제대로 이루어지려면 적어도 100년 앞은 내다보고 계획을 세워야 한다고 합니다. 그런데 우리나라에서 시행되고 있는 모든 계획은 불과 5년 앞을 보지 못하고 있는 것 같습니다.
얼마전 세워진 상계동 신시가지, 우리나라 최대규모라고 말하는 이 신시가지에 기초공사를 시작한지는 불과 5년도 되지 않습니다. 그당시 불과 5년 앞을 내다보지 못하여 이제 1/3정도 밖에 입주하지 않았는 데도 벌써부터 심각한 교통체증 현상을 보이고 있읍니다. 말로만 빛 좋은 개살구꼴이 되고만 것입니다.
그때당시만 해도 앞으로 나타날 심각한 현상들이 예견되었지만 시행과정에서 무시되어 버리고 만 것입니다.
이와 마찬가지로 국내에 컴퓨터가 처음들어와 각급학교에 교육용으로 설치된 것도 불과 몇년전의 일입니다.
그당시만해도 컴퓨터란 용어자체가 생소할 때였고 관심을 가진 사람들도 그리 많지 않았읍니다. 그런데 원대한 계획과 포부를 가지고 시작하였던 각급학교의 컴퓨터교육이 불과 5년도 되지 않아 흐지부지 되어 버리고 말았읍니다.
학생들을 가르칠 교사가 부족하고, 각 기종마다의 호환성이 없으며, 제대로 활용할 소프트웨어가 부족해서 등등 여러가지 이유로 도중하차하고 말았읍니다. 학생들 조차도 단순한 놀이기구 정도로 생각하는 층도 있읍니다. 아예 어떤 학교에서는 고장 등을 이유로 제대로 사용하지 못하게 하는 경우도 있다고 합니다.
그후 서서히 식어가는 컴퓨터 열기에 다시 불이 붙어 컴퓨터에 대한 관심이 집중되고 있읍니다.
2000년대 정보화사회에 대비 전국 초·중·고학생 964만명에 컴퓨터 교육을 실시할 것이며 학교당 PC 31대를 보급하고 교육용 소프트웨어는 총 857종을 개발보급 활용하며 이를 위해서 90년부터 각급학교의 정규과목에 편성되도록 한다는 문교부의 방침이 발표되자 벌써부터 2000년대의 우리들 자신들을 그려보는 사람들도 있읍니다.
중학교는 올해부터, 국민학교와 일반계 고등학교의 경우 90년부터 시행될 각급학교의 컴퓨터교육 시행에 앞서 대두된 몇가지 문제점들을 집고 넘어가고자 합니다.
첫째, 초·중학교에는 8비트 기종을, 고등학교에는 16비트 기종을 보급하기로 이미 확정하였으나 이 자체는 상당히 고려해 보아야 할 문제라고 생각됩니다. 왜냐하면 지난번에 시행한 컴퓨터 교육에서의 시행착오를 다시 겪지 않기 위해서는 기종간의 호환성 문제와 앞으로 설치될 국가전산망과 관련해서 이들과 연계해서 사용할 수 있는 기종이 선택되어야 한다고 생각합니다.
둘째, 컴퓨터 교육담당교사 대책으로 우선 '88년 말까지 학교당 2명 배치했으며 매년 3~6천명씩, 기존교사를 재교육한다고 하지만 지금까지는 대부분의 교사가 단기연수를 통해 배출되고 있으며 실업계 고등학교 교사는 대부분이 자격연수를 받는데 비해 일반계 고등학교는 단기연수를 받은 사람도 극소수인 점을 감안하면 문제점이 상당히 심각하리라고 생각됩니다. 특히 컴퓨터교사의 양성문제는 양적인 확보보다는 질적인 보충이 시급히 이루어져야 되지 않을까 합니다.
지금까지는 교사의 수준에 따라 강의내용이 달라지는 경우가 비일비재하였읍니다.
앞으로는 이런 점이 보완되어야 학생들이 제대로 배울 수 있지 않을까 생각합니다.
이상과 같이 먼 장래를 위해서는 이러한 문제점들이 신중히 검토되어 최선의 방법이 선택되어야 할 것이며 부족한 점이 있다면 반드시 보완되어야 한다고 생각됩니다.

MS-DOS 입문 ②

디렉토리 구조

박승제/여주컴퓨터학원장

계층적 디렉토리

DOS는 디스크내의 자료를 화일단위로 관리한다는 것은 지난 2월호에서 설명한 바 있다. 따라서 DOS는 디스크의 특별한 영역에 화일의 제목, 확장자, 크기, 만들어진 날짜, 속성(특성 : Attribute) 등을 수록하여 두는데 이 부분을 디렉토리(Directory) 영역이라고 부른다.

이 디렉토리에는 화일에 대한 기초적인 정보가 수록되어 있으므로 프로그램을 실행하는데 있어서 DOS는 이곳을 참조하게 된다. 마치 책의 첫머리에 '차례'를 참조함으로써 원하는 부분을 바로 찾을 수 있는 이치와 같다.

2S2D(양면 배밀도) 플로피디스크의 경우에는 사용자가 사용할 수 있는 기억용량은 모두 360KB(1KB=1024바이트)이며, 최대 112개의 화일을 저장할 수 있다. 화일의 크기가 거대하다면 단 몇개만 저장해도 기억용량이 모자라겠지만 화일의 크기가 작아 디스크에 사용할 공간이 많이 남아 있다 하더라도 112개 이상의 화일은 저장할 수가 없다. 왜냐하면 DOS를 설계할 당시 디렉토리의 공간을 최대 112개의 화일에 대한 정보만을 수록할 수 있도록 했기 때문이다.

그러나 요즈음 많이 보급되고 있는 하드디스크는 기억용량이 10M(1Mega=1024KB), 20M, 40M 등으로 플로피디스크에 비하여 기억용량이 엄청나게 크다. 따라서 취급할 수 있는 화일의 수는 당연히 많아져 수백개에서 수천개의 화일을 수록할 수 있다. 화일의 수가 많다보면 당연히 화일을 관리하기가 어려워진다. 수천개의 화일속에서 자신이 원하는 화일을 찾는 과정을 생각해보라. 더우기 화일의 이름만 보고 그 내용이 무엇인지 짐작한다는 것은 화일의 수가 증가하게 되면 도저히 불가능할 것이다.

예를 들어 굉장히 두꺼운 백과사전의 경우를 상상해 보자. 차례만 해도 수십페이지에 이르는 아주 두꺼운 그 책에서 '철'이라는 항목을 찾을 경우, 차례를 뒤적여 원하는 항목을 발견하는데 많은 시간이 걸리게 된다. 또 애써 찾아간 항목이 '제발 이젠 철 좀 들어라'의 '철'이었다면 우리가 찾고자 하는 '쇠'라는 의미의 철을 찾기 위해 다시 차례를 뒤적여야 한다.

만약 이 책의 차례가 의학, 금속, 공예…등의 큰 제목만 나와 있고 세부적인 차례는 그 항목에 다시 마련되어 있다면 사용자는 먼저 '금속'의 위치를 찾은 후, '금속'에 기록되어 있는 차례를 뒤적여 '철'이란 항목을 찾아가게 될 것이다. 그렇게 된다면 이름이 같아서 생기는 혼란도 없을 뿐더러 같은 종류의 항목을 관리하는데 많은 도움을 주게 된다.

DOS에서도 화일을 이런 식으로 관리하고 있는데 이런 방법을 '계층적인 디렉토리'라고 부른다.

계층적 디렉토리는 하드디스크에서 그 진가를 발휘한다. 예를 들어 김, 이, 박 세 사람이 컴퓨터를 공동으로 사용한다고 할 때 하드디스크 상에 KIM, LEE, PARK 등으로 디렉토리를 따로 설정해 두고 각각의 디렉토리에 자기의 화일을 수록하도록 한다면 컴퓨터는 디렉토리단위로 화일을 관리하므로 화일의 이름이 같더라도 전혀 별개의 장소에 저장을 하게 된다.

이것은 각자 자신의 디렉토리만을 사용하게 함으로써 마치 각자가 하드디스크의 전부를 사용하는 것과 같은 효과를 기대할 수 있다. 이때의 디렉토리구성을 그림으로 나타내면 **그림 1**과 같다.

KIM, LEE, PARK 등의 디렉토리들은 모두 루트디렉토리에 속해 있으며 각 디렉토리는 서로 독립적으로 운용된다. 루트디렉토리 입장에서는 KIM, LEE, PARK 등의 서브디렉토리가 '자 디렉토리(Sub directory)'가 되며, 서브디렉토리 입장에서는 루트디렉토리가 '부 디렉토리'가 된다. 서브디렉토리는 각각 또다른 서브디렉토리를 가질 수 있으며 이 디렉토리들이 계속 발전해 나가면 **그림 2**와 같은 트리(tree) 구조가 된다.

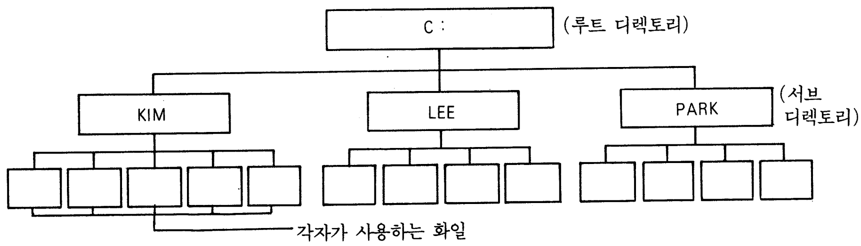
서브디렉토리 설정

[형식] MKDIR '디렉토리명'

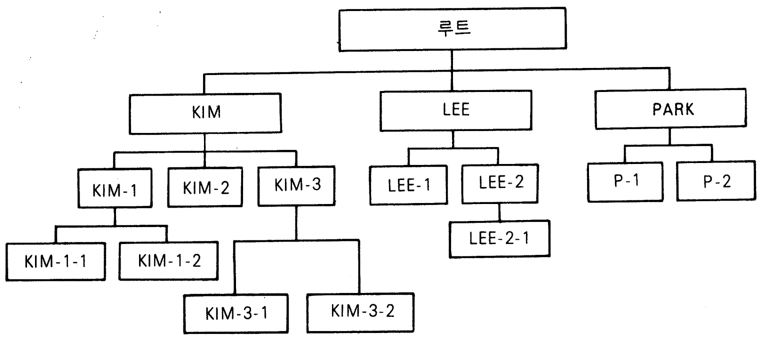
디렉토리명은 공백이 없는 8자 이내의 문자열로 구성할 수 있으며 'MKDIR' 명령은 약어로 'MD'라고 쓸 수 있다.

화면 1에 서브디렉토리를 만드는 과정을 보였다. 서브디렉토리가 설정되어 있는 경우 DIR명령을 내리면 화면 1처럼 디렉토리명 다음에 '〈DIR〉'을

〔그림 1〕 디렉토리의 분할



〔그림 2〕 디렉토리의 트리구조



〔화면 1〕서브디렉토리의 작성

```
C:\>MD KIM
C:\>MD LEE
C:\>MD PARK
C:\>DIR

 Volume in drive C has no label
 Directory of  C:\

COMMAND  COM    25307   3-17-87  12:00p
CONFIG   SYS       20   1-01-80  12:00a
ANSI     SYS     1811   2-12-88  10:00a
PRTDRV   SYS    42546   2-12-88  10:00a
RAMDISK  SYS     1396  12-29-83   1:28p
KIM         <DIR>       1-25-89  12:02a ─┐
LEE         <DIR>       1-25-89  12:02a  ├서브디렉토리
PARK        <DIR>       1-25-89  12:02a ─┘
        8 File(s)    444928 bytes free
```

표시해서 일반 화일과 구별하게 된다. 이 디렉토리는 일반 화일과는 구별되므로 디렉토리명과 화일명이 같아도 무방하다.

디렉토리간의 이동

[형식] CHDIR '디렉토리명'

CHDIR은 줄여서 'CD'로 표기할 수 있다. 서브디렉토리에서 DIR명령을 내리면 화면 2와 같이 '.'을 표시한 디렉토리가 표시되는데 첫째줄에 있는 것은 루트디렉토리를 의미하여 둘째줄에 있는 것은 커런트디렉토리(Current Directory)를 의미하는 것이다.

디렉토리간의 이동은 CD명령에 의해 자신의 서브디렉토리로는 자유롭게 이동할 수 있지만 루트디렉토리나 다른 서브디렉토리로의 이동은 반드시 디렉토리의 경로(Path)를 설정하여야 가능하도록 되어 있다.

경로의 지정

경로의 지정은 '\'로 나타낸다. '\'는 역슬래시 혹은 백슬래시(Back Slash)라고 부르는데 각국의 화폐기호로 대신하도록 약속되어 있다. 그러므로 우리나라에서 생산된 컴퓨터는 '\'문자 대신에 '₩'로 대신할 수 있다.

만약 그림 2의 KIM-1-2디렉토리에서 LEE-1 디렉토리로 이동하고 싶다면 CD\LEE\LEE-1 ↵ 이라고 명령하여야 한다. 이것은 다음과 같은 의미다.

```
CD \ LEE \ LEE-1
   │       └ LEE-1 디렉토리가 LEE에 속해있음을
   │         나타낸다.
   └ LEE 디렉토리가 루트에 속해있음을 나타낸다.
```

또, KIM-1-2 디렉토리에서 KIM-1로 이동하려 한다면 다음과 같이 경로를 설정하게 된다.

```
CD\KIM\KIM-1
```

위의 예에서 알 수 있듯이 경로의 설정은 항상 루트디렉토리부터 하게 된다. 그러므로 어느 디렉토리에 있든지 'CD\ ↵'를 입력하면 바로 루트디렉토리로 가게 된다.

디렉토리의 화일관리

DIR, COPY등의 명령이 디렉토리간에 행해질 경우에는 명령어 다음에 반드시 경로를 표시해 주어야 한다. 이를 테면 KIM 디렉토리의 내용을 화면에 표시하려할 경우에는 'DIR\KIM ↵'

을 입력하여야 한다. 또한 LEE-1에 있는 COMMAND.COM을 KIM-1-2로 복사하려 한다면 다음과 같이 입력하면 된다.

```
COPY \LEE\LEE-1\COMMAND.COM \KIM
                    \KIM-1\KIM-1-2
```

화일을 삭제하려는 경우에도 반드시 경로를 설정해 주어야만 한다.

디렉토리의 삭제

[형식] RMDIR '디렉토리명'

컴퓨터로 작업을 하다보면 디렉토리를 제거할 필요가 생기기 마련이다. 디렉토리의 삭제는 RMDIR명령을 이용하며 약어로 흔히 'RD'만을 입력한다. 디렉토리가 삭제되기 위해서는 반드시 비어있어야 한다. 비어있다는 의미는 화일이 한개도 없다는 의미이다. 만약 화일이 있는데도 RD명령을 내린다면 'Invalid path, not direct or directory not empty'라는 에러가 발생하게 된다. 디렉토리가 경로설정이 잘못 되었거나, 디렉토리의 이름이 틀렸거나, 디렉토리가 비어있지 않다는 의미의 에러이다.

또한 디렉토리의 삭제는 상위 디렉토리에서 하위 디렉토리를 삭제할 수만 있고 자신 스스로는 삭제할 수가 없다. 화면 3에 디렉토리를 삭제하는 과정을 보였으니 참고하기 바란다.

서치패스의 설정

DOS에서 프로그램 등을 실행할 경우에 경로(Path)를 지정하지 않는다면 컴퓨터는 현재의 디렉토리(Current Directory)만을 검색하게 된다. 이때 실행하려는 화일이 존재하지 않는다면 DOS는 'Bad command or file not found' 를 출력하게 될 것이다.

서치패스(Search Path)는 커런트디렉토리에서 화일이 발견되지 않을 경우 자동적으로 지정한 디렉토리를 뒤

〔화면 2〕 서브디렉토리의 출력

```
C:\>CD KIM
C:\KIM>
C:\KIM\>DIR

 Volume in drive C has no label
 Directory of C:\KIM

.           <DIR>       1-25-89   12:02a →루트디렉토리
..          <DIR>       1-25-89   12:02a →커런트디렉토리
KIM-1       <DIR>       1-25-89   12:03a ┐ 서브디렉토리
KIM-2       <DIR>       1-25-89   12:04a ┘
ASSIGN   COM     1561   3-17-87   12:00p ┐
BASICA   COM    36403   3-17-87   12:00p │ 화일들
LABEL    COM     2377   3-17-87   12:00p ┘
       7 File(s)    400384 bytes free
```

〔화면 3〕 서브디렉토리의 삭제

```
C:\>CD KIM
C:\KIM>
C:\KIM\>DIR

 Volume in drive C has no label
 Directory of C:\KIM

.           <DIR>       1-25-89   12:02a
..          <DIR>       1-25-89   12:02a
KIM-1       <DIR>       1-25-89   12:02a
KIM-2       <DIR>       1-25-89   12:02a
KIM-3       <DIR>       1-25-89   12:25a
       5 File(s)   460288 bytes free


C:\KIM>RD KIM-1       ← KIM-1 디렉토리를 삭제한다.
Invalid path, not directory, or directory not empty   ← 서브디렉토리에 화일이 있다는 메시지


C:\KIM>DEL KIM-1\*.*   ← KIM-1 디렉토리 화일을 모두 삭제한다.
Are you sure (Y/N)?Y

C:\KIM>RD KIM-1        ←서브디렉토리를 삭제한다.
C:\KIM>           ←삭제되었다.
C:\KIM>DEL KIM-2\*.*   ←KIM-2에 있는 화일을 모두 지운다.
Are you sure (Y/N)?Y
C:\KIM>RD KIM-2        ←KIM-2를 삭제한다.
C:\KIM>DIR

 Volume in drive C has no label
 Directory of C:\KIM
```

```
.          <DIR>    1-25-89   12:02a ┐
..         <DIR>    1-25-89   12:02a ├ KIM-1,KIM-2가 삭제되었다.
KIM-3      <DIR>    1-25-89   12:02a ┘
      3 File(s)  473088 bytes free
```

져보도록 하는 명령으로 아주 유용하게 쓰인다. 이를테면 KIM 디렉토리에서 LEE 디렉토리에 있는 화일을 자주 실행할 필요가 있다면 다음과 같이 설정한다.

```
PATH \LEE
```

이렇게 설정해 두면 프로그램을 실행할 경우 현재의 디렉토리에서 지정한 화일이 발견되지 않는다면 자동적으로 LEE디렉토리에 있는지를 검사한 후 있으면 실행하게 된다.

서치패스는 한번 설정해 두면 다음에 다시 설정하거나 전원이 다운될 때까지는 유용하다.

앞에서와 같이 서치패스를 설정해 두면 LEE 디렉토리에 있는 프로그램은 현재의 디렉토리에서 경로의 설정없이도 자유롭게 실행할 수 있다. 서치패스는 다른 드라이브 사이에도 설정할 수 있다.

즉 PATH A : ↵ 에 의해 컴퓨터는 자동적으로 A : 드라이브를 검색하게 된다. 또한 복수의 디렉토리에 대한 서치패스도 설정할 수 있다. 이를테면 현재의 디렉토리에서 화일이 발견되지 않는 경우 LEE와 PARK 디렉토리를 차례로 검색하게 하고 싶다면

```
PATH \LEE;\PARK
```

과 같이 ';'으로 디렉토리를 구분하여 설정하면 된다.

또 서치패스가 어떻게 설정되어 있는지를 알고싶다면 PATH ↵만을 입력하면 된다.

그러면 컴퓨터는 'PATH=\ LEE'와 같이 현재의 서치패스를 표시하여 줄 것이다.

디렉토리 위치의 자동표시

디렉토리가 복잡하게 구성되어 있는 하드디스크에서 작업을 하다 보면 현재 내가 어느 디렉토리에 와 있는지 잊어버리는 경우가 왕왕 있게 된다. DIR 명령을 내리면 현재의 디렉토리를 표시해주긴 하지만 이건 상당히 불편하다. 이때는 그냥 CD ↵ 만을 입력하면 된다. 그러나 이 경우도 일일이 입력을 해야 하므로 불편하기는 마찬가지이다.

DOS에는 프롬프트(PROMPT)라는 명령이 있는데 이것은 'A>', 'C>' 등과 같이 출력되는 DOS 프롬프트를 사용자가 마음대로 바꿀 수 있도록 해 주는 명령어이다.

다음과 같이 입력해 보자.

```
PROMPT $P$G
```

이 명령에 의해 DOS의 프롬프트는 커런트디렉토리의 경로를 프롬프트로 설정하게 된다. 즉, 현재 KIM-1에 있다면 'C>' 대신에 'C : \KIM\KIM-1' >으로 표시된다. 하드디스크의 사용자라면 이 명령이 아주 유용하게 쓰여질 것이다.

프롬프트의 변경

프롬프트명령은 표 1의 인수에 의해 다양하게 출력된다. 표 1에 정의되지 않는 문자열은 그대로 화면에 출력된다.

'PROMPT Hellow! $P-$G ↵ 를 입력하면 'Hellow! C : \KIM\KIM-1 >'를 출력한다.

이것은 다음과 같은 의미이다.

〔표 1〕 프롬프트 정의문자

문 자	기 능
$	이 이하의 문자 앞에 항상 붙인다.
T	현재의 시간
D	현재의 날짜
P	현재의 드라이브와 경로표시
V	버전(Version) 번호.
N	커런트 드라이브
G	'>' 문자
L	'<' 문자
B	':' 문자
_(밑줄)	줄을 바꿈(Line Feed)
S	공백 출력
E	ESC 문자(아스키코드 27)

```
PROMPT Hellow!  $P - $G
   │             │ │ └── '>'를 출력
   │             │ └──── 화면에 그대로 출력(공백 포함)
   │             └────── 현재의 드라이브와 경로 출력
   └──────────────────── 화면에 그대로 출력(공백 포함)
```

PROMPT ↵ 만을 입력한다면 프롬프트는 초기화되어 커런트드라이브만을 표시하게 된다.

하드디스크의 화일관리

하드디스크에는 여러개의 화일과 디렉토리가 공존하므로 화일과 디렉토리가 섞여 지저분해지기 쉽다. 특히 루트디렉토리가 이런 형태가 되기 십상이다. 화일을 효율적으로 관리하기 위해서는 적절한 서브디렉토리를 설정해야 한다.

루트디렉토리에는 서브디렉토리만 존재하도록 하고 일반적인 화일은 해당 서브디렉토리에 저장하도록 하는 것이 좋다. 그러나 루트디렉토리에 단 하나의 화일도 없다면 그 컴퓨터는 C : 드라이브만으로는 부팅하지 못한다.

모든 화일은 서브디렉토리에 저장하되 COMMAND.COM과 확장자가 .SYS인 화일은 반드시 루트디렉토리상에 두도록 한다. COMMAND.COM은 DOS의 골격을 이루는 화일로서 컴퓨터가

부팅될 때 읽혀져야 하므로 당연히 루트디렉토리에 있어야 한다. 또한 확장자명이 SYS인 화일은 컴퓨터가 부팅될 때 참조하는 화일이므로 이것 역시 루트디렉토리에 존재해야 한다.

그 외의 화일은 모두 해당 디렉토리에 저장하되 적절히 서치패스를 설정하여 사용한다면 화일을 효율적으로 관리할 수 있을 것이다.

배치화일의 작성

컴퓨터를 사용하다 보면 좋든 싫든 간에 키보드를 두들겨서 명령을 내리게 된다. 이때 입력해야할 내용이 많고 이 또한 매번 같은 내용을 입력해야 한다면 짜증이 날 것이다.

키보드로부터 입력할 일련의 내용들을 화일로 작성해 두고 컴퓨터로 하여금 그 화일을 읽어가면서 스스로 수행하게 한다면 사용자의 수고는 한결 덜 수 있을 것이다.

MS-DOS는 이런 일련의 절차를 배치화일(Batch File)이라는 이름으로 관리하게 된다. '배치(Batch)'라는 단어의 뜻이 '다발, 묶음'등이니 만큼 배치화일은 명령어를 수록해둔 화일로 볼 수 있다.

배치화일은 에디터나 워드프로세서 등을 이용하여 작성하지만 그 내용이 간단한 것이라면 DOS상에서도 직접 작성할 수 있다.

하드디스크의 사용자라면 컴퓨터를 켠 후 자신의 디렉토리로 가기 위해서 매번 다음과 같이 입력할 것이다.

```
CD\KIM\KIM-1
PROMPT $P -$G
PATH C:\;\LEE
DIR
```

이때 앞의 내용들을 배치화일로 작성해 두면 그 화일명만 입력하므로써 앞의 내용들을 키보드에서 모두 입력한 것과 똑같은 처리를 하게 된다. 배치화일을 작성하기 위해서는 다음과 같이 입력한다.

```
C>COPY CON KIM.BAT
CD\KIM\KIM-1
PROMPT $P -$G
PATH C:\;\LEE
DIR
<ctrl>+<Z>
```

'〈Ctrl〉+〈Z〉'는 화일이 끝나는 것을 의미한다. 'CON'은 Console의 약자로서 여기서는 키보드를 의미한다. 'COPY CON KIM.BAT'라는 명령은 키보드에서 직접 입력받아 'KIM.BAT'라는 화일을 작성하라는 뜻이다. 이때부터 입력한 명령들은 컴퓨터가 실행하는 대신 디스크에 수록하게 된다. 컴퓨터는 실제로 〈Ctrl〉+〈Z〉가 입력된 이후에 한꺼번에 저장한다. 이때 디스크상에 'KIM.BAT'가 존재한다면 먼저의 내용은 없어지고 새로 입력한 내용으로 바뀌게 된다. 또 확장자가 '.BAT'인 화일은 '.COM'이나 '.EXE' 화일보다 실행 순위가 낮으므로 디스크에 'KIM.COM'이나 'KIM.EXE'가 존재하면 'KIM.BAT'는 실행시킬 수 없게 된다. 따라서 배치화일에 이름을 붙일 때는 이런 점을 고려하여 기억하기 쉬운 이름으로 하는 것이 좋다.

화면 4에서 보듯이 'KIM'만 입력하면 그 다음은 컴퓨터가 배치화일을 읽어가면서 실행해 준다. 이때 배치화일의 내용(명령)들이 마치 키보드에서 입력되는 것처럼 화면에 표시되는 것을 볼 수 있다.

그러나 배치화일의 명령어가 화면에 표시되는 것은 별로 보기에 좋지 않다. 따라서 배치화일속의 명령어를 화면에 표시하지 않고 실행시키려면 배치화일의 첫머리에 'ECHO OFF'를 기술한다. 'ECHO'는 화일속의 명령어 등을 화면에 표시하느냐(ON) 안하느냐(OFF)를 나타낸다.

'ECHO OFF'는 그 이후의 명령에 대해서 화면에 표시하지 않는 것이므로 'ECHO OFF'자체는 화면에 출력되게 된다. 이 명령 자체도 화면에 표시하지 않으려면 DOS버전이 3.3이라면 '@'를 'ECHO OFF'앞에 붙이면 된다. '@'는 그 직후의 코맨드에 대해서 화면 출력을 중지시키는 명령이다.

버전이 3.3이전일 때는 이런 기능이

〔화면 4〕 배치화일의 실행

```
C:\>KIM                  ←키보드에서 입력한다.
C:\>CD\KIM\KIM-1         ┐
C:\>PROMPT $P-$G         │컴퓨터가 화일의 내용을 키보드에서 입력한 것처럼
C:\KIM\KIM-1→PATH C:\;\LEE│  실행한다.
C:\KIM\KIM-1→DIR         ┘

 Volume in drive C has no label
 Directory of C:\KIM\KIM-1

.            <DIR>      1-25-89   12:21a
..           <DIR>      1-25-89   12:21a
ASSIGN   COM     1561   3-17-87   12:00p
CONFIG   SYS       20   1-01-80   12:00a
BASICA   COM    36403   3-17-87   12:00p
LABEL    COM     2377   3-17-87   12:00p
COMMAND  COM    25307   3-17-87   12:00p
PCTOOLS  EXE    83840   9-14-85    1:19p
        8 File(s)     168960 bytes free
```

없으므로 'CLS'의 코맨드를 이용한다. 'ECHO OFF' 다음 줄에 'CLS'를 기술하면, 'ECHO OFF'가 출력되는 순간에 화면을 지워버려 화면에 아무것도 출력되지 않는 효과를 얻을 수 있다. 배치화일을 보기 좋게 하려면 다음과 같이 한다.

```
C>COPY CON KIM.BAT
ECHO OFF
CLS
CD\KIM\KIM-1
PROMPT $P -$G
PATH C:\;\LEE
DIR
<ctrl>+<Z>
```

배치화일은 되도록 루트디렉토리에 두는 것이 좋다.

자동실행화일의 작성

앞의 배치화일은 'KIM'만 입력하면 자동적으로 일련의 절차를 실행해 주지만 그나마 입력하기 귀찮은 사람들을 위해 DOS는 '자동실행화일'이라는 것을 지원해준다.

자동실행화일은 컴퓨터가 부팅되면 스스로 읽어들여서 일련의 절차들을 수행하므로, 게으른 사람들을 위한 최대의 배려이다. 자동배치화일은 앞에서 설명한 배치화일과 작성하는 방법이 똑같다. 단지 화일이름이 'AUTOEXEC. BAT'인 것만 다르다.

만약 독자들이 어떤 디스크를 가지고 있다면 'DIR'명령으로 그 디스크의 디렉토리를 살펴보기 바란다. 디스크에 'AUTOEXEC.BAT'화일이 존재한다면 그 디스크를 드라이브에 넣고 컴퓨터를 켜는 것만으로 대부분 프로그램이 스스로 수행된다.

하드디스크의 경우에는 여러 사람이 사용하고 또, 사용하는 사람마다 사용하는 패키지가 다를 것이므로 자동실행화일은 컴퓨터의 초기설정을 하는 것으로 그쳐야 한다. 하드디스크의 사용자라면 자동실행화일은 다음과 같이 설정할 수 있을 것이다.

```
C>COPY CON AUTOEXEC.BAT
ECHO OFF
CLS
DATE
PROMPT $P -$G
PATH C:\;\LEE
DIR *.BAT
<ctrl>+<Z>
```

화일내용의 표시

DOS상태에서 화일의 내용을 읽어볼 경우에는 TYPE문을 이용한다.

[형식] TYPE [드라이브:] 화일명.확장자

TYPE문은 모든 화일의 내용을 표시해 주는 것은 아니고 아스키(ASCII) 형식으로 저장된 화일만 가능하다. 아스키형식으로 수록되지 않는 화일을 TYPE문으로 읽어보면 화면에 이상한 문자들만 찍히게 될 것이다.비아스키(Non-ASCII) 형식의 화일을 2진(Binary) 화일이라고 하며 대부분의 프로그램 화일이 여기에 속한다. 아스키형식의 화일은 확장자가 '.BAT'이거나 '.DOC" 혹은 확장자가 없는 화일들이 여기에 속할 수 있다. 각자 앞에서 작성한 배치화일을 TYPE문으로 읽어보기 바란다.

만일 독자들이 새로 응용프로그램을 구입하였다면 DIR로 화일의 디렉토리를 살펴보기 바란다. 아마 'README. DOC' 혹은 'README'라는 화일이 있을지도 모른다. 이 화일들은 프로그램의 사용자를 위해 메이커측에서 작성해둔 것으로 그 프로그램의 사용법과 주의 사항을 수록해둔 것이므로 반드시 TYPE문으로 읽어보는 것이 좋다. 만약 그 디스크에 'README.COM'이 존재한다면 'README.COM'을 실행하도록 하라. 이 'README.COM'은 'README'라는 화일을 화면에 페이지별로 표시해주는 프로그램일 것이다.

TYPE문으로 화일의 내용을 화면에 표시할 때 화면표시를 잠시 멈추고 싶으면 〈Ctrl〉+〈S〉를 누른다. 이때 〈Ctrl〉+〈S〉를 누르는 시간이 늦은 경우에는 화면이 그냥 위로 올라가게 된다.

이에 대해 DOS에는 화면의 출력을 페이지단위로 출력하도록 지원하는 기능이 있다. 바로 'MORE.COM'이 이 기능을 수행하게 된다.

즉 TYPE 화일명 :MORE↵ 에 의해 화면에 23줄이 가득차면 화면 밑에 '---More---'라는 메시지를 출력하며 멈추게 된다. 이때 아무키나 누르면 다음 페이지가 계속된다. ':MORE'명령을 TYPE문에 이용할 경우에는 커런트드라이브의 디스켓상에 'MORE.COM'이 있어야 함은 물론이고 컴퓨터가 작업할 충분한 공간이 디스크에 있어야 한다.

화일내용의 프린터 표시

화일의 내용을 프린터에 출력하는 방법으로는 3가지가 있다. 첫번째 방법은 〈Ctrl〉+〈P〉를 입력해서 프린터로 에코(Echo)하는 방법이다. 〈Ctrl〉+〈P〉는 화면에 한줄 찍는 명령이란 것을 지난 2월호에서 설명한 바 있다.

〈Ctrl〉+〈P〉에 의한 프린터의 출력은 〈Ctrl〉+〈P〉를 다시 입력하면 중지한다.

(예) C>TYPE AUTOEXEC.BAT<ctrl>+<P>

두번째 방법은 '>PRN'을 기술하여 출력하는 장치를 바꾸는 방법이다. 이때는 화면에 출력될 내용이 모두 프린터로 찍히게 된다. 물론 이때는 화면에 아무 것도 나오지 않는다.

(예) C>TYPE AUTOEXEC.BAT>PRN

세번째 방법은 DOS 디스켓에 들어있는 'PRINT.COM'이라는 프로그램을 이용하는 방법이다. 두번째 방법의 경

우에는 출력결과는 모두 프린터에만 찍히게 되고, 컴퓨터는 프린터의 출력을 위해 전력투구(?)할 것이므로 프린터에 출력이 끝날 때까지 사용자는 컴퓨터를 활용할 수 없다.

또한 컴퓨터는 키보드의 입력도 받지 않을 뿐더러 화면에 커서조차 나타나지 않는다. 프린터로 출력할 내용이 많다면 컴퓨터의 사용자는 출력이 완전히 끝날 때까지 10분이고 20분이고 기다리는 수 밖에 별다른 도리가 없다.

이런 경우에 'PRINT.COM'이라는 프로그램이 그 진가를 발휘한다. 'PRINT. COM'은 프린터에 출력하는 중이라도 컴퓨터를 사용 가능한 상태로 유지해 준다. 그러므로 프린터는 프린터대로 출력하면서 컴퓨터는 컴퓨터대로 사용이 가능하게 된다.

물론 이 경우에 컴퓨터는 다른 일을 처리해 가면서 프린터쪽으로 데이터를 조금씩 보내는 일을 교대로 반복하는 것이지만 사용자의 입장에서 보면 프린터와 컴퓨터가 각각 따로 동작하는 것처럼 여겨지게 된다.

이렇게 프린터와 컴퓨터의 중간에서 [illegible] 장치를 독립적으로 운용할 수 있도록 하는 프로그램을 '프린터 스풀러(Spooler)'라고 하며, 하드웨어적으로 [illegible]처리하는 장치로 '프린터 버퍼'라는 [illegible]이 시판되고 있는데 이것을 이용하면 컴퓨터와 프린터를 독립적으로 실행시킬 수 있다.

'PRINT. COM'의 실행방법은 다음과 같다.

[형식] PRINT 화일명 화일명....

프린터에 출력할 화일들이 여러개인 경우에는 한개 이상의 공백으로 분리하여 계속 나열하면 된다.

'PRINT. COM'의 실행시 줄 수 있는 옵션 및 파라미터(Parameter)는 다음 기회에 'DOS'명령어 요약'시 설명할 예정이므로 우선은 이렇게만 알고 있어도 된다.

디스크의 볼륨레이블

디스크 외부에는 그 디스크의 명칭을 기록하여 붙이게 된다. 이를 레이블이라고 하는데 이 레이블은 인간이 보기 위함이지 컴퓨터의 입장에서 보면 이 레이블은 전혀 의미가 없다. 디스크에 컴퓨터가 알 수 있는 레이블을 부여할 경우에는 'LABEL.COM'을 이용한다.

[형식] LABEL 레이블명

레이블명은 '.'을 포함하지 않는 11자 이내의 문자열로 구성될 수 있다. 이때 공백은 포함해도 된다. 디스크에 볼륨레이블을 붙여두면 'DIR'등의 명령으로 디렉토리를 표시할 때 화면상에 나타나게 된다. 화면 5에 볼륨레이블을 붙이는 과정을 보였다.

레이블은 디스크를 포맷할 당시에 부여할 수도 있다. FORMAT B:/V↵에 의해 포맷이 끝나면 레이블을 물어오게 된다. 이때 적절히 입력하면 볼륨레이블이 붙게 된다. '/V'는 볼륨레이블을 붙이라는 의미의 스위치이다.

화면상의 그림을 프린터로 출력

화면을 있는 그대로 프린터에 출력하는 경우에는 〈Shift〉+〈Print Screen〉키를 누르면 된다는 것은 지난 2월호에서 설명한 바 있다. 또한 화면을 프린터에 그대로 옮기는 것을 하드카피(Hard Copy)라고 부른다는 것도 잘 알고 있을 것이다.

화면에 그림이 그려진 상태(게임 등의 경우)에는 하드카피를 실행하면 프린터에는 이상한 문자들만 가득 찍히게 된다. 화면의 그래픽을 프린터로 찍을려는 경우에는 DOS 디스켓에 들어있는 'GRAPHICS.COM'을 이용한다.

컴퓨터를 켠 후 'GRAPHICS.COM'을 실행시키면 화면에는 아무 것도 나타나지 않지만 〈Shift〉+〈Print Screen〉키가 눌려지면 화면의 그림을 프린터에 찍어준다. 'GRAPHICS.COM'은 한번 실행시켜 놓으면 전원을 끌 때까지

〔화면 5〕 볼륨레이블의 작성

```
A:\>DIR/W

 Volume in drive A has no label
 Directory of A:\        └디스크에 볼륨레이블이 없다.

ASSIGN   COM   CONFIG   SYS   BASICA   COM   LABEL    COM   COMMAND  COM
PCTOOLS  EXE   DATE     DAT   DATENOW  COM   DEBUG    COM   FORMAT   COM
GRAPHICS COM   HCOPY    COM   HGKEY    COM   MAGIC    COM   ANSI     SYS
PRINT    COM   QUICK    COM   RAMDISK  SYS   C-KILLER COM   SETUP    COM
KIM            CG       COM   ABC      BAT
        23 File(s)    80896 bytes free

A:\>LABEL A:  ←키보드에서 입력
Volume in drive A has no label

Volume label (11 characters, ENTER for none)? MS-DOS 3.20
                                              └잘못 입력하였다.
Invalid characters in volume lable
Volume lable (11 characters, ENTER for none)? MS-DOS 3-20
                                              └→ 다시 입력하였다.
A:\>
```

그 효력을 발휘한다.

프린터의 조정

프린터는 80컬럼용과 132컬럼용이 있다. 132컬럼용의 프린터를 가지고 있더라도 'TYPE'문 등으로 프린터에 출력하면 80문자를 찍은 후에 자동적으로 줄을 바꾸게 된다. 이것은 DOS가 프린터를 80컬럼으로 인식하기 때문이다. 프린터를 132컬럼으로 세트하고 싶을 때에는 DOS 디스켓에 있는 'MODE.COM'을 이용한다.

〔형식 1〕 MODE LPT1:132[,6,P]
([ ]안은 생략할 수 있음을 의미한다)

'LPT1:'은 프린터가 '시리얼 포트(Serial Port)1'에 연결되어 있음을 나타내는 것이다. IBM-PC는 최대 3개까지 프린터포트를 장착할 수 있는데 각각 LPT1:~LPT3:까지 구분된다. 대부분의 경우 프린터는 포트1에 접속한다.

위의 '132'는 132컬럼으로 세트하라는 의미이며, '6'은 프린터 행간의 간격을 1/6인치로 하라는 의미이다.

프린터는 초기설정이 1/6인치로 되어 있으며 이것은 연속용지 1페이지에 최대 66줄을 출력할 수 있다. 16절지의 경우에는 종이의 크기에 따라 다소 다르지만 61~63줄이 찍힌다.

프린터의 간격을 1/8인치로 설정하려면 '6'대신 '8'을 기술한다. 1/8인치로 설정하면 페이지당 88줄을 출력할 수 있지만(16절지의 경우 78줄) 이때는 프린터의 행 간격이 없게 된다.

일반적인 보고서나 서류의 경우 1/8인치로 설정한 다음 1줄 찍고 1줄 건너뛰고(skip)하는 식(Double space)으로 출력하게 되면 행의 간격이 적당히 벌어져 보기좋게 된다. 화면 6에 1/6인치와 1/8인치의 경우를 수록하였으니 참고하기 바란다. 'MODE.COM'을 실행하기 위해서는 반드시 프린터가 켜져 있어야 한다.

〔화면 6〕

```
REM This message is the showing MODE program.
REM This mode is setted MODE LPT1:132[,6,P]
REM ..............1989.2.x Personal Computer.
```

(a) 1/6인치인 경우

```
REM This message is the showing MODE program.
REM This mode is setted MODE LPT1:132[,8,P]
REM ..............1989.2.x Personal Computer.
```

(b) 1/8인치인 경우

'P'는 'Device Timeout'에러 발생시에도 프린터를 쓸 수 있도록 해주는 파라미터이다.

프린터는 3, 4개의 스위치가 붙어있는데 그 중 'Ready' 혹은 'Select'라는 스위치를 누르면 프린터의 램프에 파란 불이 들어오며, 다시 누르면 불이 꺼진다.

불이 켜진 상태는 프린터가 사용 가능한 상태이며, 불이 꺼져있으면 프린터에 출력할 수 없게 된다. 프린터의 'Ready' 램프에 불이 들어오지 않으면 DOS는 불이 켜지기를('Ready' 스위치를 눌러 주기를) 수초간 기다리게 된다.

이때 스위치를 눌러주면 프린터가 정상적으로 동작하지만, 수초간 기다려도 'Ready'상태로 되지 않으면 DOS는 프린터에 문제가 발생한 것으로 간주하고 'Device Timeout'에러를 발생시킨 후 프린터출력을 중단하게 된다.

만일 프린터출력시 종이를 갈아 끼우기 위하여 고의로 정지시킨 경우라면 시간이 다소 걸리더라도 에러가 발생하지 말아야 할 것이다. 이럴 경우를 대비해서 설정한 옵션이 바로 'P'파라미터이다.

시스템디스켓의 작성

컴퓨터는 항상 DOS 디스크로부터 부팅한다. 디스크가 부팅되고 DOS의 프롬프트가 나온 후에야 디스크의 교환도 가능하며, 원하는 프로그램의 실행도 가능하게 된다.

하드디스크의 사용자라면 디스켓을 교환할 필요가 없이 앞에서 배운 'CD' 명령만으로도 디스켓을 교환하는 효과를 얻겠지만 대부분의 플로피디스크의 사용자들은 자주 디스켓을 교환해야 하는 불편을 겪게 된다. 또한 프로그램을 DOS 디스켓에 수록하고 자동실행화일을 만들면 되겠지만 DOS 디스켓에 프로그램을 수록할 공간이 없다면 디스크를 교환하여야 한다.

그러나 디스켓에 컴퓨터를 부팅할 수 있는 최소한의 기능만을 수록하고, 이 디스켓에 응용프로그램을 카피한 다음, 자동실행화일을 작성하면 이 디스켓을 넣고 컴퓨터를 켤 경우 스스로 부팅하며 응용프로그램이 실행될 것이다.

이때 컴퓨터가 부팅될 수 있는 디스크를 시스템(System) 디스크라고 하는데 이 디스크를 만들 경우에는 'FORMAT. COM'을 이용한다. FORMAT B:/S↵ 에 의해 B:드라이브의 디스크는 포맷이 되고, 그 이후에 컴퓨터의 부팅에 필요한 화일을 컴퓨터가 수록하게 된다. '/S'는 디스크를 포맷한 후에 시스템용 화일을 복사하라는 의미이다. 앞에서 배운 '/V'로 볼륨도 붙이려면 'FORMAT B:/S/V'라고 입력한다.

'DIR'명령으로 이 디스크의 디렉토리를 살펴보면 'COMMAND.COM'이 수록되어 있는 것을 발견할 수 있다. 실제로는 'COMMAND.COM'외에도 'IBMBIO.SYS'와 'IBMDOS.SYS'라는 화일이 수록되어 있지만 이 두 화일은 DIR명령으로는 나타나지 않을 뿐만 아니라 DEL명령에 의해서도 지워지지 않는다. DOS에서는 이 두 화일을 '숨겨진 화일'이라고 부른다.

디스크를 포맷할 경우에 '/S'를 붙이는 것을 잊었을 경우에는 DOS 디스켓에 있는 'SYS.COM'을 이용하게 된다. SYS B: ↲ 에 의해서 DOS는 시스템용 화일(숨겨진 화일)을 B:드라이브로 전송한 다음 'System Transferred' 라는 메시지를 출력하게 된다. 이 때는 'COMMAND.COM'은 전송되지 않으므로 COPY명령으로 'COMMAND.COM'을 따로 복사한다.

'/S'명령으로 시스템을 전송하려는 경우 전송받는 디스크는 반드시 비어있어야 한다. 다른 화일이 수록된 디스크에 'SYS'명령을 실행하면 'No room for system'이라는 메시지가 출력되며 시스템의 전송은 이루어지지 않는다. 이때는 디스크를 다시 포맷하여야 한다.

시스템용으로 디스크를 포맷하면 약 65,000바이트가 소요된다. 그러므로 데이터를 수록하기 위한 디스크를 '/S'를 붙여서 포맷하면 기억공간을 그만큼 손실하는 결과를 초래하게 된다.

〈용어해설〉

●**화일(File)**
어떠한 목적에 따라 조직적으로 수집된 정보나 기록의 모임을 말한다. 예를 들어 자기드럼이나 자기테이프, 자기디스크, 자기버블(Bubble) 등의 매체에 기계코드로 기억되어 있는 것이 화일이다.

●**레코드(Record)**
계산기와 입출력장치나 외부기억장치 사이의 데이터 전송 및 기억의 최소단위이다. 이러한 레코드의 집합을 화일이라고 한다. 그리고 레코드를 종합해서 보내는 전송방법을 블록전송이라고 한다.

●**디렉토리(Directory)**
시스템이 갖고 있는 화일의 일람표로서 화일관리의 핵심이 된다. 화일의 명칭과 용량, 일자 등을 기억한다.

●**트리구조(Tree Structure)**
목구조라고도 하며 보통 짜여진 구조의 원리가 나무와 같은 형태로 이루어져 트리구조 또는 목구조라고 부른다. 즉 나무의 가지가 데이터가 지나다니는 선이고, 가지가 모여 있는 부분을 마디(Node)라고 하며, 들어오는 가지가 없고 나가는 가지만 있는 마디를 뿌리(Root), 그리고 가지가 들어가기만 하고 나오지 않는 마디를 잎이라고 한다. 이 트리구조는 데이터, 프로그램 등의 구조를 그래프로 표현할 때 사용한다.

●**패스(Path)**
어떤 마디에서 다른 마디까지의 경로를 말한다.

●**서치(Search : 탐색)**
대량의 데이터중에서 필요한 특성을 지닌 데이터를 찾아내는 것으로 데이블 탐색 또는 데이블 룩업(Table Lookup)이라고도 한다. 서치효율의 극대화는 목적 데이터를 찾는 과정에서 화일의 데이터를 몇번 조회했느냐에 따라서 결정된다.

●**프롬프트(Prompt)**
OS가 오퍼레이터에게 주는 메시지로서 키로부터의 입력이 가능하다는 것을 알리는 것이다.

●**프롬프팅(Prompting)**
시분할이 이루어지는 시스템에 있어서 계속해서 시분할 처리를 하기위해 유저에게 필요한 오퍼랜드를 입력하도록 요구하여 유저가 원하는 처리를 할 수 있게 하는 기능이다.

●**시분할(Time Sharing)**
하나의 장치를 둘이상의 목적에 사용하기 위해 시간을 나누어 두가지 모두 실행할 수 있도록 동작시키는 것을 말하는데, 이들은 빠른 속도로 이루어지기 때문에 유저는 동시동작이 이루어지는 것처럼 느낀다.

●**볼륨(Volume)**
하나의 단위로서 장착하거나 제거할 수 있는 기억매체를 말한다.

●**배치처리(Batch Processing)**
데이터처리에서 필요한 데이터를 모아 두었다가 그것을 일괄처리하는 방식

●**콘솔(Console)**
제어용 키, 스위치, 지시기 등을 갖추고 오퍼레이터가 필요에 따라 계산기에 동작에 개입하거나 이것을 감시하기 위해 마련된 터미널이다.

●**스풀(Spoole)**
종이테이프 등을 감는 틀을 말한다.

●**스풀링(Spooling)**
일의 실행과 사후의 처리나 출력조작을 병행하는 편리한 형식이다. 예를 들어 보조장치로부터 입력스트링을 읽거나 쓰는 일이 이 스풀링동작의 일종이다.

●**하드카피(Hard Copy)**
보고서, 리스트, 요약문 등을 사람이 눈으로 보고 읽을 수 있는 형태로 프린터 등을 이용하여 그대로 출력하는 것을 말한다. 반면에 CRT 스크린상에 마찬가지로 요약문과 같은 것을 출력시키는 것은 소프트카피(Soft Copy)라고 부른다.

●**섹터(Sector)**
1개의 트랙을 몇개로 분할한 블록으로서 레코드라고도 불리며, 읽기 / 쓰기는 섹터단위로 이루어진다.

첨단서적

컴퓨터 전문서적 통신판매 써비스

표준 MS DOS 핸드북	기전연구사	7,000원	알기 쉬운 dBASE Ⅲ	영진	6,000원	SIDEKICK plus	성안당	6,000원
MS DOS 핸드북	정보시대	3,000원	dBASE Ⅲ plus 프로그램 모음집	영진	9,000원	GW BASIC 3.2	성안당	4,500원
MS DOS 입문	성안당	5,000원	dBASE Ⅲ plus	크라운	7,000원	GW BASIC	크라운	5,500원
MS DOS	성안당	5,000원	dBASE Ⅲ plus	기전연구사	6,000원	IBM BASIC	성안당	6,000원
IBM DOS 3.2	크라운	5,000원	Auto CAD 2.6	영진	15,000원	워드 퍼펙트	우진출판사	7,000원
MS DOS 3.3	크라운	6,000원	Auto CAD 실무	크라운	5,500원	PC TOOLS DELUXE	크라운	3,000원
알기 쉬운 MS DOS	영진	5,000원	Auto CAD 실습과 응용	영진	10,000원	터보파스칼 4.0	크라운	8,000원
고급과정 MS DOS	크라운	6,000원	C언어 기초+α	교학사	8,000원	WORDSTAR 4.0	우진	6,000원
MS DOS 핸드북	크라운	9,000원	Run C	교학사	7,000원	WORDSTAR 실무	영진	4,500원
DOS 3.3	영진	8,000원	C언어의 이론 응용	성안당	7,000원	애플 Ⅱ 입문	영진	3,500원
MS DOS 4.0	연학사	6,500원	C 프로그래밍	동일	5,000원	Symphony 메뉴얼	크라운	7,000원
MS DOS 메뉴얼	크라운	4,000원	C언어 500제	크라운	5,000원	Symphony 메뉴얼	영진	7,000원
MS DOS와 GW BASIC	연학사	5,500원	C언어 입문과 실습	크라운	5,500원	RM COBOL	성안당	5,000원
Lotus 123 기본	크라운	5,000원	C 라이브러리	영진	5,000원	MS COBOL	성안당	6,000원
실무 Lotus	크라운	6,000원	TURBO C(User's)	크라운	5,000원	IBM PC 어셈블리	기전연구사	6,000원
Lotus Leal 2	크라운	5,000원	TURBO C(Reference)	크라운	5,000원	IBM PC 하드웨어 종합분석	성안당	5,500원
완벽 Lotus	영진	5,000원	MS C 교과서	성안당	6,000원	MSX DOS 입문	연학사	5,500원
알기 쉬운 Lotus	크라운	9,000원	Clipper Compiler	영진	5,000원	MSX DOS 슈퍼 핸드북	우진	4,500원
Lotus 123	기전연구사	6,000원	실무자를 위한 Crosstalk	영진	5,500원	MSX BASIC 컴퓨터교실	기전연구사	4,000원
실무 Lotus 123	영진	10,000원	노턴 유틸리티	영진	3,500원	컴퓨터 용어사전	영진	4,500원
실무 Lotus 123(제2장)	영진	12,000원	노턴 유틸리티	크라운	4,000원	컴퓨터 용어사전	대은	9,000원
dBASE Ⅱ	크라운	4,000원	XENIX V 메뉴얼	영진	7,000원	미니탭 메뉴얼	크라운	6,000원
dBASE Ⅲ	기전연구사	5,500원	XENIX 오퍼레이팅 시스템	크라운	5,500원	APPLE Ⅱ 종합분석	성안당	5,000원
실무 dBASE Ⅲ plus	영진	6,000원	Dr. HALO Ⅱ	교학사	3,000원	APPLE BASIC 프로그램 실습	영진	6,000원
초보자를 위한 dBASE Ⅲ plus	영진	6,000원	Dr. HALO Ⅲ	교학사	3,000원	인공지능	영진	4,000원
프로그래머를 위한 dBASE Ⅲ plus	영진	6,000원	센서 인터페이싱	기전연구사	24,000원	마이크로프로세서	동일	7,000원

※ 본지에 게재된 기타 신간서적도 주문 가능함

- **주문방법** : 직접 구입 또는 교재명과 출판사명을 기재하여 해당금액에 발송료 500원을 추가하여 우체국 소액환으로 동봉 신청하면 즉시 우송함.
- **보 낼 곳** : ㊐140-012 서울시 용산구 한강로2가 전자단지 17동 3층 가열 325호
  전화 : 716-6024

전문인을 섬기는 **첨 단 서 적**

둘리가 본 컴퓨터세계 ②

컴퓨터 그는 누구인가?

이만희/한국교육개발원 컴퓨터교육 연구센터

하나, 둘, 셋 그리고 네번째 일어난 사건

인간의 역사를 이야기할 때 우리는 흔히 도구의 사용, 불의 사용, 산업혁명을 중요한 발전의 시점으로 삼는다. 그러나 그런 것 말고도 바로 눈앞에 보이지는 않지만 인간을 동물의 세계에서 벗어나게 한 중요한 것이 숨어있다고 생각한다.

"안다는 것"

인간은 스스로 누구인가를 알려고 노력했고, 도구의 사용방법이나 불의 사용법을 알고자 노력했다. 다시 말해 알고자하는 인간의 역사가 정보의 역사일 수 있다. 컴퓨터는 정보의 역사에 중요한 혁명을 일으키고 있다.

"컴퓨터 그는 누구인가?"

정보의 역사속에서 그 의미를 다시 한번 생각하고 그를 대하기를 바란다.

"컴퓨터는 정보처리의 도구이다."

오랜 인류의 역사에서 정보처리방법과 기술은 몇번의 커다란 진보가 있었고 인류역사 발전에 중요한 계기가 되었다. 이중 중요한 4가지 변화는 다음과 같다(그림 1 참조).

첫째는 말의 등장이다.

정보를 전달하는 수단으로서 말은 동물의 단순한 신호보다 아주 복잡한 정보를 효율적으로 전달할 수 있게 하였고, 정보의 효율적인 전달은 인간을 동물과 다른 존재로 만들었다.

둘째는 글자의 발명이다.

정보를 전달하기 위해 소리로써 잠시 존재하거나 인간의 두뇌안에 있던 정보를 망각하지 않도록 보관하여 다음 사람이 그 위에 다시 새로운 정보를 더하게 되는 편리한 도구로써 글자가 발명되었다. 따라서 배우고 사용하기 쉬운 글자를 사용한 종족과 국가는 계속적인 정보의 축적과 전달로 번성을 누려왔다.

세째는 인쇄기술의 발명이다.

사람의 손에 의한 글자의 기록은 제한된 양일 수 밖에 없었고 소수의 사람에게만 정보전달이 가능하였다. 인쇄기술은 글자로 기록된 정보를 몇백 몇천배로 확산시켜 수많은 사람에게 전달하였다. 산업혁명도 수많은 사람들이 정보를 소유하게 된 인쇄혁명을 바탕으로 여러정보가 복합적으로 적용되어 가능했다고 할 수 있다.

네째는 컴퓨터의 발명이다.

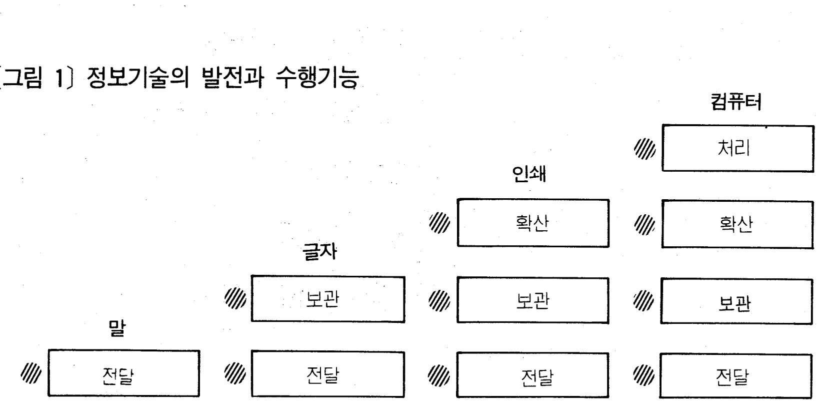
컴퓨터에 정보를 보관하고, 보관된 자료를 찾아내는 작업은 인간의 두뇌 안에서만 가능한 일이었으나 컴퓨터는 이를 몇천 몇만배 빠른 속도로 대신하고 있다.

사람이 컴퓨터를 이용하게 된 이후 정보의 생산을 획기적으로 증대시키고 있다. 이는 컴퓨터를 사용한 정보처리가 아주 간편해지고, 또 빠른 속도로 진행되고 있기 때문이다. 예로서 인쇄혁명 이후에도 책의 출판은 많은 전문인력과 경비가 요구되어 능력있는 소수의 사람만이 책을 출판할 수 있었다. 그러나 컴퓨터를 이용한 워드프로세서가 등장하자 대부분의 사람들이 적은 비용으로 직접 인쇄하거나 책을 쉽게 만들어 낼 수 있게 되었다. 심지어는 국민학생도 자신의 일기를 컴퓨터로 인쇄하여 보급할 수 있게 된 것이다.

이는 지식(정보)의 생산이 제한된 소수에서 다수로 변화함에 따라 정보의 생산 역시 폭발적으로 늘고 있음을 의미한다. 과거의 역사에서 정보처리방법과 기술이 진보된 민족과 국가가 번영했듯이 지금 우리가 맞고 있는 새로운 기술-컴퓨터의 혁명-에서도 이 같은 현상이 나타날 것이다.

정보처리기술의 낙후는 다시 우리에게 고난과 퇴보는 물론 역사에서 완전한 도태를 의미할 수 있다.

〔그림 1〕 정보기술의 발전과 수행기능



컴퓨터 족보와 가족

1 컴퓨터의 족보

(1) 조상의 전설

원시시대에는 셈을 위해 손가락이나 돌멩이가 사용되었고, 그 뒤 문명이 점차 발달하면서 수판을 만들어 사용하였다.

1642년 파스칼은 톱니바퀴를 이용한 기계계산기를 만들었고, 진공관이나 트랜지스터가 만든 전자시대 이전에는 자동계산기를 만들려는 노력이 있었다.

산업혁명이 일어나기 시작했던 1800년대 유럽에서는 직조기로 생산하는 옷감의 무늬를 짜기 위해 여러가지 배열로 구멍이 뚫린 나무카드를 사용하였다. 나무판에 있는 구멍의 배열 상태에 따라 지나가는 실의 배열이 바뀌고 옷감의 무늬도 변하였다. 다시말해 나무카드를 바꾸어 주면 옷감의 무늬를 바꾸어 짜는 편리한 직조기계가 만들어 졌다. 이것은 초보적이지만 컴퓨터의 프로그램원리와 동일한 것이다.

1835년 영국의 베비지는 이의 원리를 더욱 발전시켜 해석기관 즉, 컴퓨터를 만들려고 노력하였으나 기계장치의 기술적인 한계로 실패하였다. 1944년에 와서야 전기 릴레이를 사용한 "전기기계식 계산기" MARK-1이 만들어져 실용화에 성공하였다. 이것이 지난호에 설명한 컴퓨터해부학 시간에서의 '디버그'라는 언어를 유래하게 한 것이다.

1946년 드디어 진공관을 이용한 "전자계산기" ENIAC이 탄생하였다. 진공관을 사용하면 과거의 기계장치와 달리 실제로 움직이는 부품이 없어지고 전자펄스(신호)로 점멸하는 스위치(플립플롭)를 만들 수 있다. 컴퓨터가 1초에 30만킬로미터의 속도로 움직이는 전자의 속도를 이용하게 되자 성능에서 혁명적인 발전이 이루어졌다.

(2) 세대차이

인간은 수만년에 걸쳐 문명을 진보시키기도 하고, 혹은 퇴보시키기도 하면서 발전했지만, 컴퓨터는 불과 수십년 전에 등장하자마자 십년 사이에 몇백 몇천배의 능력이 증가되는 변화를 겪었다. 더구나 크기는 몇천분의 1로 작아져서 하나의 IC 안에 들어가는 컴퓨터도 만들고 있다.

컴퓨터는 보통 10년에 한번씩 큰 발전이 있어 이를 세대구분의 기준으로 하고 있다. 컴퓨터에서 한세대의 차이는 너무나 커서 실로 십년에 강산이 변하는 정도가 아니라 천지개벽이 되는 정도의 변화를 이룬다. 그 변화의 방향은 보다 빠르게, 보다 작게, 보다 쉽게 이렇게 세가지점에서 발전하고 있다(표 1 참조).

컴퓨터는 1946년 미국에서 진공관을 이용한 제1세대 기종이 개발된 이후 트랜지스터를 이용한 제2세대와 IC(집적회로)를 이용한 제3세대를 거쳐, 현재에는 VLSI(Very Large Scale Integration) 를 응용한 제4세대 컴퓨터가 사용되고 있으며 제5세대의 컴퓨터가 실험 연구되고 있다.

소프트웨어는 컴퓨터마다 다른 종류의 기계어를 사용하다가 모든 컴퓨터가 공통적으로 사용하는 고급언어가 개발되었다. 오퍼레이팅 시스템은 컴퓨터가 보다 효율적으로 주변장치와 스스로를 운영할 수 있게 해주었으며, 현재는 컴퓨터와 다른 컴퓨터간에 통신과 자료교환을 할 수 있는 네트워크 시대에 와있다. 지금 활발히 연구되고 있는 인공지능이 실현되면 논리적 추론이나 자율적 판단이 가능한 인간의 두뇌에 가까운 기능을 갖게 될 것이다.

2. 컴퓨터 가족 현황

통상 컴퓨터라고 부르는 것중에는 작은 타자기만한 것부터 큰 방에 가득 차는 커다란 것도 있다. 또 혼자서 쓰는 것부터 수십명이 복잡한 과학계산을 할 수 있는 것도 있다.

요즘은 퍼스널컴퓨터가 널리 사용되며 그 능력이 이전 같으면 메인프레임이라고 불리울 정도의 강력한 것이 나오고 있다.

또 컴퓨터의 발전속도가 너무 빨라서 몇년만 지나면 분류자체가 무의미해지는 정도이나 컴퓨터를 일반적으로 분류하는 기준으로 보면 다음과 같다 (표 2 참조).

표 2와 같이 컴퓨터와의 종류를 규모에 따라 다섯가지로 분류하고 있으며, 용도는 다음과 같다.

- **퍼스널컴퓨터** : 개인이 간단한 사무처리용으로 주로 사용
- **마이크로컴퓨터** : 소규모 회사에서 여러 사람이 소규모 사무처리용으로 사용
- **수퍼 미니** : 대규모 사무처리용이나 중규모 과학계산용으로 사용
- **메인프레임** : 대규모 과학계산용이나 대모 시스템 관리에 사용(예 : 은행)
- **수퍼 컴퓨터** : 메인프레임보다 처리속도

〔표 1〕 컴퓨터의 발전과정

년도	50년대 1946~1955	60년대 1955~1965	70년대 1965~1975	80년대 1975~	90년대 1990~
세대분류	제1세대	제2세대	제3세대	제4세대	제5세대
하드웨어 혁신기술	진공관	트랜지스터	IC 집적회로	VLSI 대규모 집적회로	광(빛) 및 차세대소자
처리속도 (1초당)	1,000	100,000	10,000,000	100,000,000	? 초고속 병렬처리
소프트웨어 신기술	원시 기계어 (컴퓨터마다 독자적 언어)	고급언어 사용 (COBO FORTRAN) 공통언어	오퍼레이팅 시스템 발달 (O/S)	데이터베이스, 컴퓨터 네트워크 활용	인공 지능 소프트웨어

〔표 2〕 컴퓨터의 분류

구 분	가격(원)	사용자수	통상호칭
퍼스널컴퓨터	수십~수백만	1	소 형
수퍼 마이크로	수백~수천만	5~32	
수퍼 미니	수천~수억	32~128	중 형
메인프레임	수억~수십억	128이상	대 형
수퍼	수십억이상		

나 기억용량이 아주 크게 요구되는 분야에 사용(예 : 기상예측).

(3) 퍼스널컴퓨터와 워크스테이션

퍼스널컴퓨터는 개인용으로 사용되는 컴퓨터의 일반적인 명칭이다. 그 처리능력은 간단한 게임을 즐길 수 있는 정도에서 메인프레임정도에 까지 이르고 있다. 특히 엔지니어링 분야나 건축설계, 컴퓨터 그래픽 등의 분야에서 생산성향상을 위해 아주 고가이더라도 강력한 컴퓨터를 혼자서 사용하기도 한다.

이것을 워크스테이션이라고 불러왔으나 요즘은 퍼스널컴퓨터와 굳이 구별하기 어렵게 되었다.

컴퓨터 S.O.S...O.S...D.O.S

컴퓨터를 배우다 보면 O.S(운영체제 : Operating System)라는 말을 자주 듣게 된다. 컴퓨터에 문제가 생겨 컴퓨터를 잘아는 사람에게 물어보면 그건 O.S의 영역이라고 밀어버리는 경우가 많다. 알듯하면서도 잘 모르는 O.S의 의미를 약간만 이해하고 O.S의 용어를 조금만 사용하면 당신도 다른 사람에게 컴퓨터 전문가처럼 보일 수 있다.

1. O.S의 탄생과 역할

(1) 컴퓨터의 인기관리

컴퓨터가 탄생된 후 그 재빠른 솜씨를 보여주자 많은 사람들이 일감을 들고 컴퓨터 앞에 몰려들었다. 즉시 컴퓨터 주인은 비싼 이 컴퓨터를 보다 효과적으로 사용할 방법으로 한가지 프로그램을 만들어 냈다. 이 프로그램의 역할은 여러가지 작업이 밀려 있을 때 컴퓨터의 여러장치들을 보다 효과적으로 사용할 수 있도록 계획을 짜고 교통정리를 하는 것이다.

예를 들면 A, B, C, D… 여러사람이 줄을 서서 프린터로 찍어 줄 것을 부탁할 때 A가 제일 일찍 부탁했어도 그 작업량이 아주 많고 C의 양이 제일 적다면 C의 것을 먼저 찍어서 보낸 후에 A의 것을 찍도록 일의 순서를 O.S가 자동적으로 정해준다. 이런 일을 사람이 일일이 순서를 정하고 스위치를 켰다면, 그 사이에 컴퓨터는 쉬고 있어야 한다. 이럴 경우 O.S는 1초도 쉬지않고 컴퓨터가 계속 일하도록 작업지시를 한다.

(2) 교통의 기본원칙

컴퓨터가 세상에 수없이 널리 퍼진 오늘날은 O.S의 역할이 약간 바뀌었다. 옛날에는 이 세상에 컴퓨터가 불과 몇 개뿐이라 파리하고 날카로운 얼굴의 컴퓨터 기술자만이 컴퓨터 주위에 모여서 사용목적에 적합한 독자적인 사용방법을 직접 개발하여 사용하였다. 그 시대는 컴퓨터가 너무나 희귀한 존재로 새로운 컴퓨터가 하나 만들어지면 그에 맞는 새로운 사용방법을 만드는 것이 당연하였다.

그러나 요즘은 거리에 다니는 자동차보다 빠른 추세로 컴퓨터가 늘고 있다. 또한 컴퓨터가 이렇게 많아지면서 일정한 규칙이 필요해졌다. 수많은 컴퓨터가 저마다 다른 사용방법을 갖고 있다면 사용하는 사람들이 어지러워진다. 즉, 거리를 달리는 수많은 자동차들이 서로 사고를 내지 않고 주행하기 위해 필요한 교통규칙과 같다. 교통규칙을 준수하면 운전자가 어떤 종류의 차를 이용하든 쉽게 목적지에 갈 수 있다. 이와 같이 동일한 O.S를 갖고 있다면 각기 다른 컴퓨터라도 사용하는 입장에서 볼때 비슷한 컴퓨터처럼 느끼게 된다. 즉, 새로운 컴퓨터가 나오더라도 사용자는 새로운 사용법을 배울 필요없이 곧바로 쓸 수 있다.

1964년 IBM이 세계 최초로 발표한 OS / 360시리즈는 컴퓨터 운영체제의 역사에서 가장 중요한 사건이었고 컴퓨터 업계에서 IBM의 위치를 확고부동한 것으로 만들었다.

(3) 섬에서 대륙으로

무인도와 같이 고립된 섬에서는 모든 것을 자급자족해야 한다. 만일 O.S가 없다면 컴퓨터의 사용자들은 이와 같은 신세가 되었을 것이다. 그러나 CP / M이나 MS-DOS 또는 UNIX와 같은 유명한 O.S용으로는 이미 수많은 프로그램들이 개발되어 산더미처럼 쌓여 있다. 사용자들은 O.S를 통해서 이 수많은 프로그램들을 곧바로 이용할 수 있다.

여러 컴퓨터 선배들이 만들어 놓은 프로그램중에는 자기가 하고 싶은 일에 꼭맞는 것이 이미 개발되어 있는 수가 많다. 보통 회사에서 하는 봉급계산이나 장부정리는 물론이고 심지어는 과일의 익은 정도에 따라 자동적으로 등급을 결정하고 분류해서 상자에 담을 수 있도록 하는 프로그램도 나와 있다.

홀로있는 고립된 섬과 같은 컴퓨터를 수없이 많은 프로그램이 쌓여 있는 소프트웨어 대륙으로 통하도록 하는 다리와 같은 일을 O.S가 하고 있다.

2. O.S의 구성

소프트웨어의 세계에서 O.S는 기본적인 역할을 한다. 눈앞에 여러가지 과일이 열리는 신기한 나무가 있다면 보

통사람들은 나뭇잎 사이에 열린 과일에만 관심을 갖기 쉽다. 달려있는 여러가지 과일은 우리가 일에 사용하는 프로그램의 결과와 같다. 그러나 겨울이 되어 나뭇잎과 과일이 다 떨어지고 남는 것은 나무이다. 진정으로 과일을 아끼는 농부라면 그 나무의 상태를 잘 파악하고 돌보아 준다. 이렇게 바로 먹을 수는 없지만 계속해서 맛있는 과일이 열리도록 해주는 역할을 컴퓨터에서는 O. S가 하고 있다.

키보드, 모니터, 디스크드라이브와 같은 하드웨어를 직접 제어하는 프로그램의 모임을 BIOS(기본입출력부 : Basic Input Output System)이라 한다. BIOS는 "휨웨어"로 처음부터 ROM에 고정하여 컴퓨터의 스위치만 켜면 곤충이나 동물이 갖고 있는 본능과 같이 곧바로 동작하도록 되어 있다(그림 2 참조).

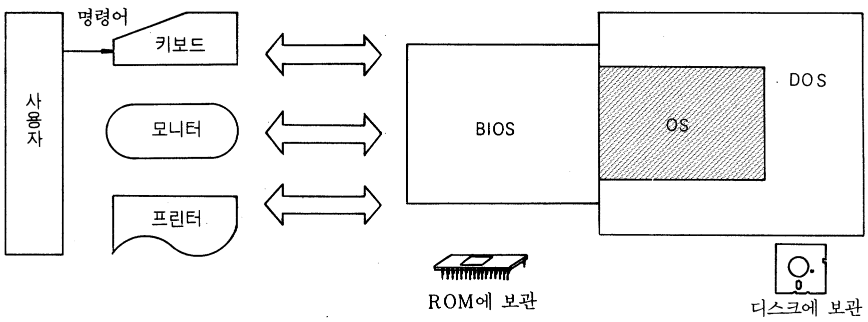
이와는 달리 O.S중에서도 디스크에 담겨 있는 부분이 있다. 모든 O.S를 전부 다 컴퓨터 메모리로 옮겨 놓으면 너무 많아서 컴퓨터를 효율적으로 사용할 수가 없다. 따라서 O.S의 대부분은 디스크에 담아서 보관하고 필요할 때만 불러서 사용한다. 이것을 DOS (디스크 운영체제 : Disk Operating System) 라 한다.

사실 과일나무 주인도 나무뿌리나 나무기둥은 함부로 손을 못대듯이 일반적인 컴퓨터 사용자가 BIOS나 시스템 등은 다루기도 어렵고 그럴 필요도 없다. 그러나 DOS를 잘 다루어야만 컴퓨터를 효과적으로 쓸 수 있다. 나무도 가지치기를 잘해야 좋은 과일이 나오는 것과 같다. DOS는 다음과 같이 세부분으로 구성되어 있다.

● **DOS본체(디스크 관리)**

DOS는 디스크에 들어 있는 O.S이지만 그 디스크 자체를 관리하는 일도 한다. 따라서 디스크 장치안에 디스크가 들어 오면 컴퓨터는 즉시 디스크안에 들어 있는 내용을 알기 위해 DOS를 읽어본다. DOS는 책의 목차와 같

〔그림 2〕 O. S의 기능



이 그 디스크에서 사용된 부분과 여백으로 남아 있는 부분을 알려주기도 한다.

● **내부 코맨드**

컴퓨터에서 자주 사용되는 명령들로 맨처음 컴퓨터를 가동시킬 때 디스크에서 컴퓨터 내부로 옮겨간다. 어느 부분을 복사하거나 지우거나 또는 화면에 보여주거나 하는 기본적인 역할을 수행하는 부분이다. 내부 코맨드의 예로서 DIR, COPY, DEL, TYPE 등이 있다.

● **외부 코맨드**

컴퓨터 메모리 내부에 항상 들어있지 않고 필요할 때만 디스크에서 불려가는 것이다. 자주 쓰지 않는 명령어들은 보통 외부 코맨드로 분류된다. 업무용 프로그램도 외부코맨드의 일종이라고 우길 수 있지만 보통의 의미로는 컴퓨터 구입시 따라오는 DOS 디스크에 들어 있는 것을 말한다(그림 3 참조).

3. O.S의 세력판도

O.S는 컴퓨터를 운영하고 프로그램을 활용하는 기본으로 컴퓨터 역사를 새로 만드는 어려움을 겪고 싶지 않다면 개발되어 있는 O.S를 사용해야만 한다. 문제는 O.S도 마치 국제간의 정치판도처럼 몇개의 그룹으로 나누어져 있다. 한 국가가 보통 자유진영 또는 공산진영에 속하고 이것도 아니면 동맹을 반대하는 동맹에라도 끼듯이 보통의 경우, 프로그램은 어느 한가지 O. S에서만 수행된다.

여기서 퍼스널 컴퓨터에서 유명한 O.S들을 살펴보기로 한다(그림 4 참조).

● **CP / M**
(Control Program for Microcomputer)

퍼스널 컴퓨터 O.S의 대부와 같다. 한때 강력한 세력을 떨쳤으나 이제는 좀 부진하다. 8비트용으로 CP / M 80, 16비트용으로 CP / M 86이 있다.

● **MSX-DOS**

8비트인 MSX 컴퓨터는 하드웨어의 사용법도 통일규격을 따른다. 16비트용 MS-DOS와 비슷한 명령어를 갖고 있다.

● **MS-DOS**
(Micro Soft-Disk Operating System)

IBM에서 사용하면서 전세계에서 가장 많이 퍼져 있다. 원류는 CP / M에서 갈라져 나왔으며, IBM에서는 PC-DOS라고 불렀다.

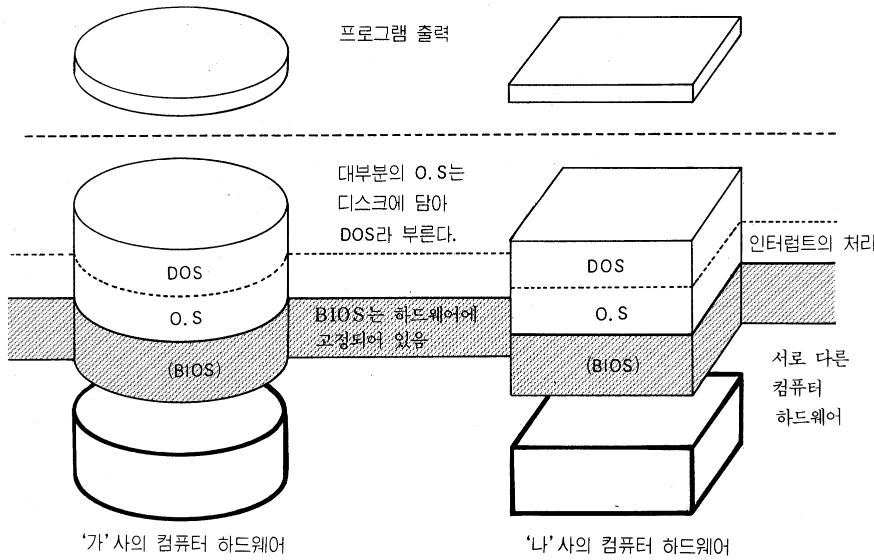
● **OS / 2**

최근 IBM에서 새로 발표한 IBM PS / 2에서 사용하는 O.S이다. 이것은 MS-DOS의 대부분의 명령을 그대로 수행하도록 하고, 멀티태스크(동시에 여러 프로그램 수행)기능과 통신기능이 강화되어 있다.

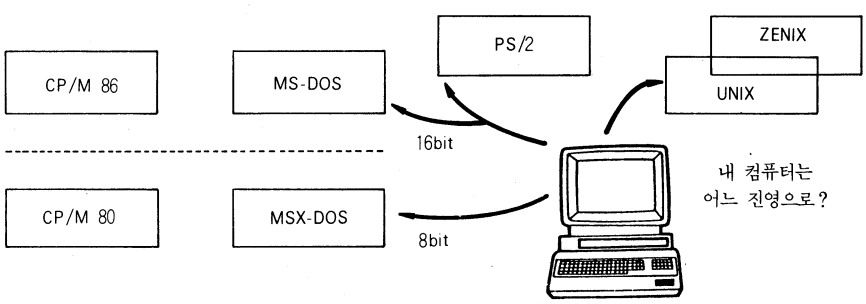
● **UNIX**

퍼스널컴퓨터 다음 세대의 O.S 왕자를 노리고 있는 강력한 도전자로 기대된다. 본래는 벨 연구소에서 미니컴퓨터의 멀티유저, 멀티태스크에 적합하도록 만들어진 것으로 최근 컴퓨터 네트워크가 강조되면서 이 기능으로 UNIX의 인기가 높아지고 있다. ZENIX등

〔그림 3〕 O. S의 구성



〔그림 4〕 O. S의 종류



여러변형으로도 존재한다.

컴퓨터에 만연하는 AIDS

1. 무고한 피해자

컴퓨터 세계에 갑자가 나타난 아주 위험한 프로그램(일명 컴퓨터 AIDS 또는 컴퓨터 바이러스)들이 컴퓨터나 디스크에 치명적인 피해를 주고 있다. 사람에게 병을 옮는 것도 아닌데 무슨 야단이냐고 강건너 불보듯 하거나 "그것보아라. 컴퓨터 좋아하다 큰 코 다치지"하며 고소해 하는 사람도 있을지 모르지만 그는 우리의 생활에서 컴퓨터가 하는 일이 얼마나 중요하고 밀접한지 잘 모르기 때문이다.

만일 은행에 예금한 돈의 컴퓨터 기록이 사라져서 한푼도 없는 것으로 된다면 수표는 부도가 나고 신용이 무너져 사회생활에서 큰 피해를 입는다. 그리고 병원에서 수술을 기다릴 때 병원의 컴퓨터 기록이 사라진다면 생명에 위협을 받을 수 있다. 또 지하철이나 비행기 역시 컴퓨터의 도움을 받아 운행되고 있는 데 갑자기 컴퓨터가 마비된다면 그 결과는 생각하기 싫어질 것이다. 이와 같이 일상생활과 관련된 많은 자료가 컴퓨터에 입력되어 디스크에 보관되어 있으므로 디스크가 파괴되면 개개인에 대한 자료 역시 파괴된다. 그런데 이런 어처구니 없는 일들이 컴퓨터 바이러스 프로그램 때문에 누구에게나 아무런 경고없이 찾아올 수 있게 되었다.

대부분의 정보가 컴퓨터로 처리되는 오늘날은 중요한 정보를 담아 놓은 디스크가 파괴되면 그 정보에 따라 중요한 판단을 하거나 그 정보에 관련있는 사람이 겪는 괴로움은 신체에 생기는 병보다도 더 크고 괴로울 수 있다. 그렇다고 컴퓨터를 사용하지 않는 것은 마치 교통사고가 무서워 차를 없애자는 사람과 같다. 때문에 싫어도 컴퓨터와 함께 지내야 하는 우리 모두를 불안하게 만드는 것이 바로 이 바이러스 프로그램이다.

이 악성 전염성 프로그램은 우연히 발생한 것이 아니라 사람이 고의적으로 만든 것이며 지금도 누구인가에 의해 더욱 치명적이고 악랄한 성질을 갖도록 개발되고 있다.

2. 근원

바이러스 프로그램이 생긴 근원중에 하나는 오래전(1980년경)부터 벌어진 소프트웨어 복사전쟁에서 찾을 수 있다. 퍼스널컴퓨터가 많이 보급되자 편리한 기능의 소프트웨어를 개발하여 상품으로 파는 사람이 나타났다. 처음에는 단순히 호기심으로 복사하여 그 소프트웨어를 단지 구경만하는 사람도 있었으나 실제 자기일에 사용하여 많은 혜택을 보면서도 개발자가 수많은 땀과 노력으로 완성한 프로그램을 아무런 보상없이 공짜로 사용하고자 하는 사람이 많았다. 이에 분노한 소프트웨어 개발자들은 컴퓨터 디스크를 관리하는 O.S의 기능을 이용하여 교묘한 방법으로 불법 복사를 막고자 노력하였다. 또 반대쪽에서는 이를 피하는 방법을 찾아내고자 하는 과정에서 수많은 아이디어, 특히 O.S 및 DOS의 변칙적인 사용법에 대한 깊은 지식들이 널리 알려졌다. 복사를 방지하는 여러 프로그램중에 한가지로 "킬러 프로록"이란 것이 있는데 이것은 불법 복사를 시도하면 그 디스크 자체를 지워버리거나 그 컴퓨터에 사용되는 또 다른 디스크마저 지워버리는 가혹한 방

법으로 무단복사를 보복했다.

3. 파괴의 원리

문제는 이렇게 발전된 기술이 아무 죄도 없는 사람들에게 피해를 주기 시작한데 있다. 본래 DOS에는 COPY 기능이 있어 한 디스크에서 다른 디스크로 자료를 쉽게 옮겨 담을 수 있고 또 그래야만 한다. 이 DOS의 기능을 약간 변경하면 사람이 명령을 주지 않아도 스스로 특정한 프로그램을 다른 디스크에 자동복제 되도록 할 수 있다. 또 DOS는 디스크 전체나 일부분을 지울 수 있는 기능도 있다. 이 두 가지를 연결시키면 다른 디스크에 스스로 복제하면서 디스크를 지우는 살아있는 해충과 같은 프로그램이 만들어 진다. 이때 바로 디스크를 지워버리면 순진한 사용자도 쉽게 이 프로그램의 존재를 눈치채고 대책을 준비하므로 여기에 한가지 아이디어를 덧붙여 프로그램이 복제된 후 일정시간을 기다리도록 하면 보다 많은 다른 디스크에 바이러스 프로그램을 복제시켜 놓고 그 디스크의 최후를 마치게 할 수 있다. 마치 전쟁시 세균을 이용해 모든 사람을 무차별 공격하듯이 이 바이러스 프로그램이 한번 세상에 퍼지면 다시 불러 모을 수가 없어 계속 널리 퍼지게 된다. 더우기 이상한 취미로 장난삼아 더 강력한 전염력과 파괴력을 가진 프로그램을 만들고 기뻐하는 이가 많아 문제는 점점 심각해지고 있다.

4. 전염경로

바이러스 프로그램은 감염 후 즉시 증세가 나타나지 않고 일정기간 잠복하면서 많은 디스크에 전염시키기 때문에 피해자 스스로도 언제 전염 됐는지를 알기 어렵다. 다만 중간에 컴퓨터를 매개체로 하여 전염되는 것은 확실하다.

●디스크를 통한 전염

컴퓨터를 사용하다 보면 디스크를 바꾸거나 다른 디스크의 자료를 복사하고 읽어야 하는 경우가 많다. 만일 감염된 디스크를 사용했다면 그 이후 사용된 디스크 대부분이 감염된다. 또 다른 사람이 컴퓨터를 사용한 후에 전원을 끄지 않고 계속 사용할 경우 앞사람의 디스크로 인해 전염될 수 있다.

그러나 아무리 지독한 바이러스 프로그램도 컴퓨터의 전원을 껏다가 켜면 다시 감염된 디스크에서 불려지지 않는 한 나타나지 않는다.

●네트워크를 통한 전염

다음에 네트워크에 대한 소개를 하겠지만 컴퓨터 네트워크를 통하면 디스크 단위로 한장씩 전염되는 것보다 광범위하게 전염될 수 있다. 컴퓨터에 연결된 전화회선을 통해 프로그램을 받아오는 경우, 자료나 프로그램만이 아닌 반갑지 않은 바이러스 프로그램이 옮겨올 수 있다. 광범위하게 펼쳐진 컴퓨터 네트워크는 사실 바이러스 침입을 방어하기가 어렵다. 물론 이미 바이러스 프로그램에 충분한 대비를 하고 있겠지만, 은행의 컴퓨터는 손님들에게 편리한 온라인 서비스를 제공하기 위해 전국적으로, 보다 넓게는 전세계 대부분의 지역에 컴퓨터 네트워크를 통해 연결되어 있다. 이중 하나의 터미널이나 또는 통신선로를 통해 무단으로 바이러스 프로그램이 끼어들 수 있다.

5. 바이러스 프로그램의 예방 및 치료방법

●확실한 프로그램의 사용

잘 알려진 경로(정당한 구매를 통한)를 통하지 않고 입수한 프로그램을 함부로 사용하거나 복사를 하지 말고 혹시 사용후에는 반드시 컴퓨터를 껐다 켠다. 보통 바이러스 프로그램은 O.S에 기생하는 것이 보통이나 바이러스를 전염시킬 목적으로 게임이나 비싼 프로그램속에 일부로 집어 넣은 것도 있다. 소프트웨어의 대륙으로 통할 수 있도록 만들어진 O.S를 통해 유용한 프로그램으로 위장한 악질 프로그램이 전염되고 있으니 주의할 수 밖에 없다.

●정기적인 백업

복잡한 방지방법을 보통사람이 전부 알 수 없고 계속 발생하는 새로운 종류의 바이러스에 계속 신경쓸 수 없다. 차라리 가끔 중요한 자료가 담긴 디스크를 복사해 두면 발병시 피해를 줄일 수 있다.

●백신 프로그램

바이러스 프로그램의 동작원리를 파악해서 이에 대응하거나 그것을 제거하는 백신 프로그램도 나오고 있다. 그러나 새로운 아이디어로 작성된 바이러스 프로그램에는 효과가 없고 새로운 백신 프로그램을 구해야 한다.

6. 도덕성의 회복

이상에서 살펴본 바이러스 프로그램은 단순히 흥미거리가 아니라 아주 심각한 문제가 되고 있다. 컴퓨터 바이러스의 출발이 단순히 소프트웨어 복사만은 아니었지만 그것 때문에 더욱 증오심을 불태운 사람들이 많았다. 우리 스스로가 함부로 다른 사람의 땀과 노력을 훔쳐(무단복사) 이용하려는 생각을 버리고 다른 사람의 자료와 정보(소프트웨어)를 존중하는 마음으로 대한다면 다른 사람도 역시 우리의 중요한 자료와 정보를 존중하고자 할 것이다.

이제는 서로 상대방을 파괴하는 전쟁이 아니라 건설과 발전의 경쟁으로 나가야 한다. 바이러스 프로그램의 존재가 원망스럽지만 이미 나타난 이상, 이제는 그를 통해서 컴퓨터 사용자들이 반성하고 또 컴퓨터의 운영원리에 대해 보다 깊은 이해가 필요하다. 아마 머지않아 바이러스 프로그램에 강력한 저항력을 갖는 컴퓨터도 나타날 것이다. 그러나 유능한 컴퓨터 사용자가 되고 싶다면 컴퓨터의 운영원칙인 O.S에 대해 올바로 알고 있어야 하고 또한 다른 사람의 노력(소프트웨어)을 존중하는 사용원칙을 지켜야만 한다. PC

배치화일

배치화일은 DOS의 중요한 기능중 하나로 잘만 활용하면 매우 편리하게 컴퓨터를 사용할 수 있다. 즉, 간단한 배치화일은 자동으로 회귀하는 기능과 컴퓨터 사용루틴을 보다쉽고 안전하게 만들어 주며 사용자 스스로가 이를 직접 작성할 수 있다.

이번 호에는 이러한 배치화일의 작성과 효과적으로 배치화일을 사용하는 방법 등을 알아본다.

배치화일이란?

배치화일은 간단히 화일의 묶음이라 말할 수 있다. 이러한 화일묶음은 워드 프로세서나 에디터 등을 사용해서 만들고 그 화일에 특별한 문자는 넣지 않는다(DOS 디스크에서의 EDLIN 에디터처럼). 배치화일을 작성하려면 COPY CON:〈filename〉.BAT와 텍스트문을 입력함에 따라 DOS 프롬프트로부터 배치화일을 만들 수 있다. 여기에서 "filename"은 지정하길 원하는 임의의 화일명이고, "텍스트"는 화일에 놓이길 원하는 하나 이상의 배치화일 명령이다. 마지막 라인까지 전부 입력한 후에는 화일종료마크(^Z), 또는 F6키를 누른 후 리턴키를 친다. 그러면 COPY명령은 화일을 만들고 DOS 프롬프트 상태로 되돌아온다. 이때 화일을 정확히 만들었는지를 확인하려면 TYPE명령을 사용하여 그 내용을 디스플레이 시켜보면 된다.

배치화일에서 중요한 제한은 확장자명이 BAT이어야 한다는 사실이다.

배치화일에 포함된 화일묶음은 표준 DOS 명령(COPY, TYPE 등)의 조합으로 이루어진다. 즉, 이들 명령은 DOS 프롬프트상태(A〉) 후에 입력한 표준 DOS명령과 흡사하지만, 배치화일의 일부로서 실행되도록 특별히 설계되어 있다.

또한, 배치화일은 우선적으로 연속된 업무를, 원하는 형태로 몇번이고 자동적으로 처리해 준다. 따라서 연속적인 업무의 경우는 배치화일을 작성하여 사용하면 지루한 업무를 수월하게 처리할 수 있다.

배치화일의 다른 잇점은 프로그램을 에러로부터 안전하게 보호한다는 점이다. 컴퓨터가 실행하는 많은 업무가 만일 잘못된다면 심각한 결과를 초래한다. 복잡한 업무가 손으로 일일이 이뤄지면 에러의 가능성은 보다 높아진다. 그렇지만 배치화일에 능숙하지 않은 유저라 할지라도 배치화일을 이용하면 업무를 수행함에 있어 자동적으로 처리할 수 있고 안전함을 보다 향상시킬 수 있다.

배치화일의 다른 사용용도는 AUTOEXEC. BAT 화일이다. DOS가 부트시킨 디스크에 이 명령을 갖는 화일을 만나면 업무를 자동적으로 수행한다.

AUTOEXEC.BAT 화일은 컴퓨터를 작동시키거나 재부트 시킬 때마다 자동적으로 어떤 상태를 세트하고 프로그램을 실행한다.

배치화일작성

이제까지 배치화일의 유용함을 확인했다. 그러나 어떻게 배치화일을 만들 것인가? 배치화일 작성에 있어 우선 .BAT란 확장자명을 갖는 배치화일은 어떻게 만들며 화일안에 무엇이 들어가야 되는지 막연할 것이다. DOS 매뉴얼의 배치명령을 찾아 볼 수 밖에 없지만, 기본적으로 꼭 알아야 할 것을 여기서 다루고자 한다.

배치화일 작성에 앞서 먼저 배치명령에 대해서 알아 보자.

일반적으로 DOS는 REM과 ECHO라는 2개의 배치화일 명령을 지원하는 화일기록을 허용하고 있으며 이것은 배치화일을 보다 읽기쉽고 유용하게 해준다.

●REM명령

REM(Remark) 명령은 배치화일에 메시지를 넣을 수 있도록 한다. 다음의 **리스트** 1처럼 REM명령을 입력해 보자.

그러면 REM명령은 로터스가 실행되기 전에 메시지를 디스플레이시켜 준다.

●ECHO명령

ECHO명령은 REM명령과 같이 동작하지만 매우 다른 기능을 갖고 있다. 즉, ECHO는 메시지를 ON이나 OFF 시키는 기능을 가지고 있다. 만일, 배치화일에 ECHO OFF를 입력하면 DOS는 모든 명령을 디스플레이시키지 않고 실행한다.

만일 REM 메시지를 가지고 ECHO 명령을 하면, 비록 ECHO OFF라 할

지라도 REM메시지는 화면에 디스플레이 된다.

다음의 리스트 2와 같은 배치화일을 실행시켜 보자.

배치화일을 실행시키면 처음 나타나는 ECHO OFF명령은 디스플레이되고, 그 다음의 REM 메시지는 디스플레이되지 않는다. 그리고 두번째의 ECHO 메시지는 디스플레이되고, LOTUS와 B:WS명령은 디스플레이되지 않는다.

이럴 경우 사용자는 화일에 대한 메시지를 읽기가 무척 어렵다. 따라서 이런 경우는 PAUSE 명령을 사용한다.

●PAUSE명령

PAUSE 명령은 어떤 파라미터 없이도 사용할 수 있고, 배치화일 실행중에 배치화일을 멈추게 하며, 다음과 같은 메시지를 디스플레이하기도 한다.

```
strike a key when ready...
```

이런 메시지는 배치화일을 다시 실행하기전에 메시지를 읽고, 올바로 디스크를 삽입하기 위한 시간적인 여유를 준다. PAUSE 명령의 사용예는 리스트 3과 같다.

●GOTO명령

GOTO 명령을 사용하면 사용자가 임의로 배치화일을 바꾸어 실행할 수 있다. 즉, 원하는 곳으로 점프하여 프로그램을 실행시킬 수 있다. GOTO 명령을 쓴 예를 리스트 4에 보인다.

리스트 4에서 주목할 것은 맨끝줄에 있는 GOTO START를 첫번째 PAUSE 명령 바로전의 :START 레이블로 보내는 것이다. 이런 변화로, 배치화일 실행을 실행시키면 LOTUS와 WS사이를 계속 반복한다. 이때, PAUSE 명령중의 하나에 도착할때마다 인터럽트 실행을 위해 CTRL-BREAK나 CTRL-C 키를 친다.

배치화일은 원하는 것을 수행하기도 하지만 드라이브B에 정말로 WS프로그램이 있는지 확인해 볼 수도 있다.

〔리스트 1〕

```
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
REM THIS SYSTEM WILL NOW RUN LOTUS AND WORDSTAR
REM BE SURE TO HAVE THE WORDSTAR DISK IN DRIVE B
LOTUS
B:WS
```

〔리스트 2〕

```
ECHO OFF
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
ECHO THIS SYSTEM WILL NOW RUN LOTUS AND WORDSTAR
ECHO BE SURE TO HAVE THE WORDSTAR DISK IN DRIVE B
REM NOW RUN LOTUS
LOTUS
REM THEN RUN WORDSTAR
B:WS
```

〔리스트 3〕

```
ECHO OFF
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
ECHO THIS SYSTEM WILL NOW RUN LOTUS AND WORDSTAR
ECHO BE SURE TO HAVE THE WORDSTAR DISK IN DRIVE B
REM STOP AND WAIT FOR THE USER
PAUSE
REM NOW RUN LOTUS
LOTUS
REM THEN RUN WORDSTAR
B:WS
```

〔리스트 4〕

```
ECHO OFF
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
ECHO THIS SYSTEM WILL NOW RUN LOTUS AND WORDSTAR
ECHO BE SURE TO HAVE THE WORDSTAR DISK IN DRIVE B
REM WE START HERE
:START
REM WAIT FOR THE USER
PAUSE
REM NOW RUN LOTUS
LOTUS
REM WAIT FOR THE USER
PAUSE
REM THEN RUN WORDSTAR
B:WS
REM START OVER AGAIN
GOTO START
```

●IF명령

IF 명령은 배치화일 안과 그 주위의 것을 테스트하고 만일 이런 테스트가 이뤄지면 배치화일 명령을 실행한다. WS프로그램이 드라이브B 디스크에 있는지 테스트해 볼 수도 있고, 만약 WS프로그램이 드라이브B에 있지 않으면 사용자에게 드라이브B에 디스크를 넣도록 지시하고 나서 다시 IF 명령을 실행시킨다.

물론, 여기에서 화일변경은 IF, GOTO, PAUSE, ECHO 명령을 사용하여 할 수 있다(리스트 5 참조).

〔리스트 5〕

```
ECHO OFF
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
ECHO THIS SYSTEM WILL NOW RUN LOTUS AND WORDSTAR
ECHO BE SURE TO HAVE THE WORDSTAR DISK IN DRIVE B
REM WE START HERE
:START
REM WAIT FOR THE USER
PAUSE
REM NOW RUN LOTUS
LOTUS
REM SEE IF WORDSTAR IS ON B DRIVE
IF NOT EXIST B:%1.COM GOTO GETWS
REM GO HERE TO RUN WORDSTAR
:WS
REM WAIT FOR THE USER
PAUSE
REM THEN RUN WORDSTAR
B:WS
REM START OVER AGAIN
GOTO START
REM COULDN'T FIND WORDSTAR
:GETWS
REM TELL THE USER TO PUT WORDSTAR IN B
ECHO PLEASE INSERT WORDSTAR DISK IN DRIVE B!
REM TRY RUNNING WORDSTAR AGAIN
GOTO WS
```

●FOR 명령

사용자가 작성한 배치화일에서 작업을 시작하기 전에 드라이브A, B의 디렉토리 리스트에 FOR명령을 사용하여 명령을 줄 수 있다. FOR 명령으로 할 수 있는 예는 **리스트 6**과 같다.

FOR명령 실행방법은 DOS가 드라이브A, B를 동작시켜 %%D 변수를 세트시키고, DOS상태 이후에 FOR명령을 수행하도록 한다.

FOR명령에서 변수는 %%에 의해 단문지를 우선으로 한다.

배치화일활용

●COMMAND.COM

COMMAND.COM 프로그램은 배치화일을 읽고 화일에 만들어 둔 과제들을 수행하기 위해 무엇을 할 것인가를 DOS 오퍼레이팅 시스템에 알려주는 역할을 한다.

```
COMMAND [D:][path-definition]
[/P][/Ccommand-name]
```

COMMAND.COM의 명령계통은 [D:][path-definition] 섹션을 통해 DOS 에 유치할 장소를 명령할 수 있다.

/P 옵션은 DOS에게 COMMAND.COM을 메모리 속에 넣어 영구보존하도록 명령하고, /C 옵션은 프로그램이 "command-name"을 실행하고 되돌아갈 수 있도록 해준다.

또한 COMMAND.COM은 DOS 주변변수들을 지원한다. 이러한 주변변수를 지급하는 유저 명령으로는 SET 명령이 있다. SET 문장형식은 SET [name=[parameter]]이다.

```
SET PATH=C:\MYSTUF;C:\DOSSTUF;A:\;B:\
```

AUTOEXEC.BAT는 날짜와 시간을 짜넣거나 적당한 디렉토리로 이동시키거나 표준 프로그램의 실행 등에 편리하게 사용할 수 있다.

즉, DOS를 기동시킬 때마다 어떤 명령을 실행시키는 작업을 하게 할 수 있다.

□ 주의

배치화일을 끝내려면 CTRL-BREAK나 CTRL-C를 사용한다.

비록 DOS가 키를 입력하는 것을 순간적으로 받아들였을지라도 재빨리 이것에 반응하지 않는 수도 있다. 그 이유는 종료 메시지가 나타나기 전에 제어가 명령 프로세스에 돌아가야만 하기 때문이다. 즉, DOS가 응답하기 전에 브레이킹 포인트에 도달해야 한다.

그러나, 마무리하기 위해 Y를 치면 실행되고 있는 배치화일의 다음 시스템이 실행되기전에 완성되지 않는 것을 고려해야 한다. DOS는 사용자가 끝내기를 원하는지 묻기위해 현재 진행중인 프로세서를 중단하는 경향이 있는데, 배치화일의 실행을 가능한 빨리 멈추기를 원하지 않는다면 CTRL-BREAK나 CTRL-C를 키를 치지 않을 것이라고 가정하기 때문이다.

DOS는 배치화일을 한번에 한줄씩 실행하고, 화일의 다음줄에 알 필요가 있을 때마다 배치화일을 읽는다.

〔리스트 6〕

```
ECHO OFF
REM XYZ OFFICE PROCEDURE 11327 - IBM PC OPERATIONS
ECHO THIS SYSTEM WILL NOW RUN LOTUS AND WORDSTAR
ECHO BE SURE TO HAVE THE WORDSTAR DISK IN DRIVE B
ECHO THE FILES YOU HAVE ONLINE ARE:
REM FOR ALL DISKS DISPLAY DIRECTORY
FOR %%D IN (A B) DO DIR %%D /W:
REM WE START HERE
:START
REM WAIT FOR THE USER
PAUSE
REM NOW RUN LOTUS
LOTUS
REM SEE IF WORDSTAR IS ON B DRIVE
IF NOT EXEIST B:WS.COM GOTO GETWS
REM GO HERE TO RUN WORDSTAR
:WS
REM WAIT FOR THE USER
PAUSE
REM THEN RUN WORDSTAR
B:WS
REM START OVER AGAIN
GOTO START
REM COULDN'T FIND WORDSTAR
:GETWS
REM TELL THE USER TO PUT WORDSTAR IN B
ECHO PLEASE INSERT WORDSTAR DISK IN DRIVE B!
REM TRY RUNNING WORDSTAR AGAIN
GOTO WS
```

〔리스트 7〕

```
REM SAVE A BACKUP COPY OF THE FILE
COPY ACCTLOG.DBF ACCTLOG.BAK
REM REMOVE THE WORKING FILE
ERASE ACCTLOG.DBF
REM CREATE A NEW FILE
DBASE ACCT
```

디렉토리를 바꾸는 명령(CD나 CH-DIR) 에 의해 배치화일에서 디렉토리를 바꿀 수 있다. 또한, 디렉토리를 체계화해서 각각의 화일형식(워드프로세싱 화일, 스프레드시트 화일 등)이 이런 화일형식을 담고 있는 디렉토리안에 놓이게 된다면 그때는 배치화일내에서 적당한 디렉토리 영역을 바꿔 쉽게 다른 형식을 처리할 수 있다.

다음의 **리스트 7**과 같은 ACCT.BAT 배치화일을 만들어 보자.

이것은 현재의 장부를 백업화일에 복사하고 다시 만드는 준비를 위해 화면을 지운다. 그리고, 자동적으로 DBASE ACCT 명령화일을 실행하도록 지시하는 매개변수로 하여금 DBASE를실행하도록 한다. 이렇게 배치화일은 누구든지 각자의 용도에 맞게 작성할 수 있는 것이다.

앞에서 설명했듯이 배치화일 FOR 명령은 각 화일을 분리해서 다루기보다는 차라리 화일그룹에서 DOS기능을 수행하도록 하고 있다. 즉 FOR 명령을 사용하면 DOS명령을 한번이상 다양하게 실행할 수 있다. 다음의 예를 보자.

```
FOR %%A IN (*0!TXT) DO TYPE %%A
```

이 명령은 순서대로 루트디렉토리에 있는 모든 .TXT 화일을 타이프한다. 그러나 FOR 명령에는 몇가지 제한이 있다. 예를 들면 다른 FOR명령을 가지고 명령을 실행할 수는 없다. 즉 변수이름(%%A)을 동일하게 하기 위해 2개의 퍼센트부호(%%)를 사용하는데, 그 이유는 DOS가 하나의 %을 매개변수 시작으로 간주하기 때문이다. 배치화일 파라미터는 FOR명령과 DOS명령에서 실행하는 것을 허용하고 있다.

```
FOR %%A IN (%1!TXT) DO TYPE %%A
```

여기에서는 찾기를 원하는 .TXT화일에 상응하는 화일명 파라미터를 갖은 배치화일을 불러낸다. 또한, 화일명 파라미터안에 있는 와일드카드 문자를 통과시킬 수도 있다. 예를 들어 ???RPT는 임의의 3개의 문자로 시작해서 RPT.TXT로 끝나는 이름을 가진 모든 화일을 TYPE하게 한다.

●NUL : 장치를 이용한 배치화일

아마도, 가장 유용한게 FOR 배치화일 명령이다. FOR 배치화일 명령은 하나의 명령내에 있는 화일그룹에서 하나의 동작을 실행하여 DOS로 보내는데 이용된다. 일예로 .TXT로 끝나는 현재의 디렉토리에 모든 화일을 나타내려면 다음과 같이 하면 된다.

```
A>FOR %A IN (*.TXT) DO TYPE %A
```

그리고 DOS는 우리가 원하지 않는 출력을 프로그램에서 제거하도록 도와주는 NUL:이란 더미장치를 제공한다.

□ 주의
NUL:장치에서 출력은 시스템에 의해 제거되고 화일내에 나타나지 않는다.

이런 기능이 DOS의 블랙홀이다. 실제로 NUL:이 도움을 주는 곳은 배치화일에서 시스템을 시작하는 AUTOEXEC. BAT 화일에 있다. 다음에 흥미있는 방식으로 NUL:장치와 재수신을 이용하는 전형적인 AUTOEXEC. BAT 화일을 살펴보자.

이 화일에서는 RAM 디스크를 셋업하고 모든 출력을 NUL:에 고정시킨다. 이 경우에는 PROKEY 키보드 매크로프로그램을 사용한 것과 같이 하고, DISKCOPY를 사용자가 입력한 것을 알지 못하도록 모든 대답을 화일에 기입한다.

ASCII 텍스트 화일을 만들 수 있는 모든 프로그램으로부터 배치화일을 만들거나 수정할 수도 있다.

베이직 프로그램이 배치화일을 수정하고 메뉴들이 어떻게 첨가되는지의 예를 살펴보면 다음과 같다(**리스트 8** 참조).

이제는 **리스트 9**와 같은 프로그램 BUILD. BAS를 한 예로 살펴보자.

리스트 9에서 BUILD 프로그램은 TASK.BAT화일을 만들고 몇개의 배치화일을 TASK.BAT로 쓴다. 그래서 GO 배치화일이 TASK를 실행하는 행에 이르면, TASK는 이제 막 TASK로 쓰여진 모든 것을 포함하게 된다. 이런 프로그램의 특징은 진행중인 배치화일을 형성하는 것을 허용하고 있으며, 행해진 모든 동적인 변화를 고려하고 있다는 것이다.

〔리스트 8〕

```
ECHO OFF
REM CLEAR THE SCREEN
CLS
REM COMFORTING MESSAGE
ECHO SETTING UP RAM DISK C:
REM DO THE WORK
SUPERDRIVE C:>NUL:
REM SET PROMPT TO SHOW CURRENT DIRECTORY
PROMPT $P$G
REM MORE COMFORT
ECHO NOW SETTING UP KEYBOARD
REM KEYBOARD MACROS SETUP
PROKEY DOS.PRO /R > NUL:
REM KEEP USER INFORMED
ECHO COPYING B: TO C:
REM COPY DISK
DISKCOPY B: C: < DCANS.TXT > NUL:
```

〔리스트 9〕

```
10  REM "THIS PROGRAM CONSTRUCTS"
11  REM "THE BATCH FILE TASK.BAT"
20  REM "A MENU CAN BE CRETED TO SELECT"
21  REM "WHAT IS PUT IN TASK.BAT"
30  OPEN "O",#1,"TASK.BAT"
40  PRINT #1,"DIR"
50  PRINT #1,"PAUSE"
60  PRINT #1,"GO"
70  CLOSE #1
80  CLS
90  SYSTEM
```

또 배치화일을 잘 이용하면 사용자에게 응답을 묻고 그에 따라 진행하는 상호작용적 배치화일을 만들 수 있다.

배치화일은 유저에게 특별한 동작이 실행되어야 하는지의 여부를 묻거나 응답에 기반을 둔 배치화일 동작과정을 변경하는 것에 대한 직접적인 준비가 되어있지 않다. 일단, 화일이 만들어지면 배치화일 IF EXIST 명령은 현재의 배치화일에서 다음에 무엇을

〔리스트 10〕

```
10   ON ERROR GOTO 500
20   KILL "ANSWER.YES"
30   KILL "ANSWER.NO"
40   A$ = INPUT$(1)
50   IF A$="Y" OR A$="y" THEN GOTO 200
60   IF A$="N" OR A$="n" THEN GOTO 300
70   GOTO 40
200  OPEN "O",#1,"ANSWER.YES"
210  WRITE #1,A$
220  CLOSE
230  SYSTEM
300  OPEN "O",#1,"ANSWER.NO"
310  WRITE #1,A$
320  CLOSE
330  SYSTEM
500  RESUME NEXT
```

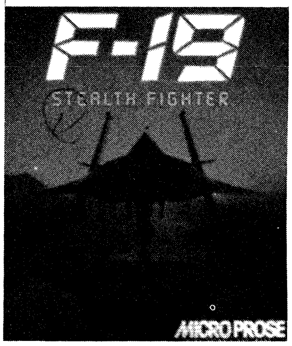
할 것인지에 대한 결론을 내린다.

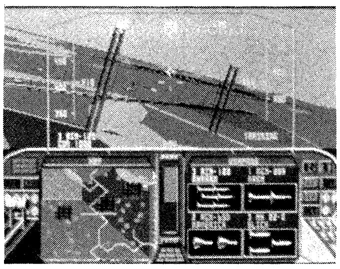
리스트 10의 YN 프로그램은 유저에게 응답을 얻기전에 화일 ANSWER. YES와 ANSWER.NO를 지운다. YN 프로그램은 N인지 Y인지를 검색하고 적합한 화일을 만든다.

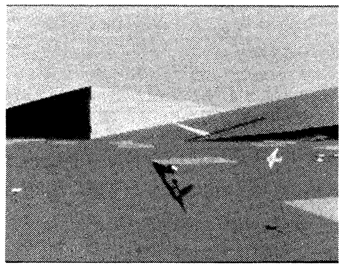
환경변수의 사용

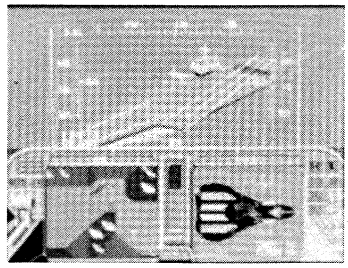
환경변수는 몇개의 TEXT를 저장할 수 있는 오퍼레이팅 시스템에서 공간이라 부르는 작은 것이다. 또한 환경변수는 DOS명령 라인, AUTOEXEC.BAT 화일, 혹은 다른 배치화일을 언급되기전에 SET되어야 한다. 각 환경변수는 전후에 %를 붙인다. 예를 들어 DOS의 PATH 값을 참조하려면 %PATH%를 입력하면 된다.

이상으로 간단하나마 배치화일을 알아봤다. 본문의 예를 직접 컴퓨터에 입력하면서 공부하면 배치화일명령어와 명령어 활용에 도움이 되리라 생각된다.







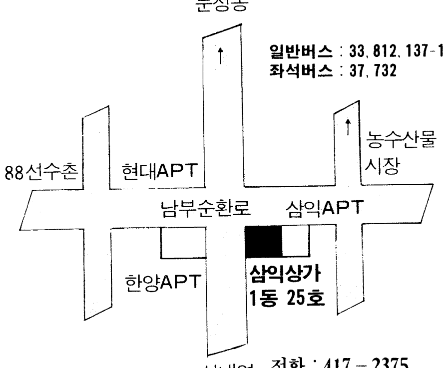


10TH FRAME PROSOWLING
221B BAKER STREET
3D HELICOPTER
ACE
ACE OF ACE
AIR TRAFFIC CONTROLLER
ALTERNATE REALITY
ANCIENT ART OF WAR AT SEA
ARTIC FOX
ATLAS
BAD STREET BRAWER
BALANCE OF POWER
BARD'S TALE I
BATTLE OF MIDWAY
BATTLE ZONE
BEYOND CASTLE WOLFENCTEIN
BIONIC COMMANDO
BLACK CAU;DRON
BLACK JACK
BLACK JACK ACAEMY
BLOCK BUSTER
BOP,N WRESTLE
BOORROWED TIME
BREACH
BUBBLE GHOST
BUCK ROGERS
BY FIRE OF SWORD
CALIFONIA GAME
CANFIELD CARD GAME
CHAMPIONSHIP BASEBALL
CHAMPIONSHIP BOXING
CHAMPIONSHIP GOLF
CHUCK YEAGER'S FLIGHT TRAINNER
COMMANDO
DALLAS QUEST
DAM BUSTER
DARK CASTLE
DEATH SWORD
DEFENDER OF THE CROWN
DESTROYER
DRAGON WORLO
EAGLE
EARL WEAVER BASE BALL
EMPIRE
F-15 STRIKE EAGLE
F-19 STEALTH FIGHTER
FALCON F-16
FAMILI FOED
FIRE POWER
FIRST EXPEDOTION
FLIGHT SIMULATOR
FLIGHT SIMUATOR V3.0
FORD SIMULATOR

FRIENDLY WARE GAME
FULL COUNT BASEBALL
GAMESTAR BASEBALL
GATO
GAUNTLET
GHOSTS'N GOBLINS
GUNSHIP
HACKER
HARD BALL
HARRIER
HEROES OF LANCE
HOSTAGE
HUNT FOR RED OCTOBER
IKARI WARRIOR
IMAGE TENNIS
INDOOR SPORTS
INFILTLARTOR
INFOCOMIC
INTO THE EAGLE'S NEST
JACK NIKLAUS GOLF
JEOPARDY
JET
JEWEL'S OF DARKESS
KARNOV
KING'S QUEST
KING'S QUEST
KING'S QUEST
KING'S QUEST
KNIGHT ORC
LAST NINJA
LEISURE SUIT LARRY II
LOCK ON
LODE RUNNER
MAD SPY VS SPY III
MAN HUNTER
MARBLE MADNESS
MASTER NINJA
MIGHT & MAGIC
MINI PUTT
MIXED UP MOTHER GOOSE
MOEBIUS
NEW SOKO BAN
NFL CHALLENGE
NINJA
OFF SHORE WARRIOR
ONE ON ONE
PATTON VS ROMMEL
PETE ROSE PENANT EFVER
PHANTASIE
PHANTASIE III
PHYCHO
PINBALL WIZARD
PINNBALL CONSTRUCTION KIT

PILICE QUEST
POLICE QUEST II
POOL OF RADIANCE
PRLEAGE SPORTS
PT-109
QUSTRON II
RAD WARRIOR
RAMPAGE
RINGS OF ZILFIN
ROADWAR 2000
SDI
SEARCH AND DISTROY
SHOGUN
SILENT SERVICE
SKATE OR DIE
SKY & AIR NAVIG
SKY RUNNER
SPACE QUEST I
SPACE QUEST II
SPIDER MAN
STAR COMMAND
STAR FLEET I
STAR FLIGHT
STAR GLIDER
SUB BATTLE
SUMMER GAME II
SUPEBYKE
SUPERSTAR ICEHOCKEY
SWORDS OF CLASS
TEED OFF CLASS
TEED OFF (GOLF GAME)
TEST DRIVE
TETRIS
THE GANES SUMMER GAME EDITION
THE LAND OF THE LOUNGE LIZARDS
THEXDER
THREE STOOGES REEL
TOP GUN
TREASURE ISLAND
TROHAN
ULTIMA I
ULTIMA II
ULTIMA III
ULTIMA
ULTIMA
WHERE IN THE EUROPE IS CARMEN
WHERE IN THE USA IS CARMEN
WHERE IN THE WOOLRD IS CARMEN
WINTER GAMES
WIZARD WARS
WIZARDRY
WORLD GAME
WORLO KARATE CHAMPIONSHIP

아이콤 약도



전화 : 417－2375
송파구 방이동 398번지
가락 삼익상가 1층 25호

회원모집 안내

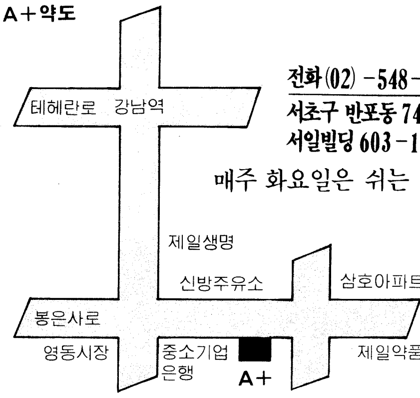
년가입비 : 10,000
월회 비 : 10,000

지방 USER안내

1. 지방회원은 송료만 추가하시면 됩니다.
2. 필요한 SOFT WARE는 소액환으로 보내주시면 프로그램을 보내드립니다.

A+약도

전화(02) －548－9717
서초구 반포동 746－5
서일빌딩 603－1호
매주 화요일은 쉬는 날입니다.



서일빌딩 603－1호(녹십자)
7층 흰건물

아카이브

정성들여 프로그램을 작성해서 그 프로그램을 디스켓에 저장하다 보면, 간혹 프로그램이 디스켓 한장의 용량을 약간 넘어설 경우가 있다. 이럴 때는 별 수 없이 두장의 디스켓에 나누어 저장해야 한다. 그리고 여러가지 프로그램들 중에 1~2장의 디스켓에 저장되는 것도 있지만 1~20장 정도의 디스켓이 필요한 유틸리티일 경우 각각의 프로그램 복사를 위해 수십번의 동일한 작업을 해야 할 때가 있다.

이러한 일들은 컴퓨터 사용자에게는 번거롭기 그지없는 작업이다. 이러한 고민을 해결하는 방법의 하나로 화일 압축 유틸리티인 아카이브(ARCHIVE)를 소개하고자 한다. 아카이브는 앞에서 설명한 문제들을 100% 해결할 수는 없지만 메모리와 디스켓을 보다 효율적으로 쓸 수 있는 방법을 제시해 주며, 그에 따르는 화일 저장의 새로운 즐거움도 맛보게 할 것이다.

아카이브의 특징은 화일을 압축(60% 정도)시키고 그 압축된 프로그램을 다시 복원시킬 수 있다는 데 있다. 또한 압축과정에서는 암호를 줄 수도 있어 중요한 프로그램을 보호할 수도 있다. 특히 통신을 하게 될 경우에 많은 양의 데이터를 전송할 때 압축시켜 보내게 된다면 압축된 만큼 시간을 절약할 수 있다. 경제적인 측면에서는 기존의 디스켓에 넣을 수 있는 용량보다 더 저장할 수 있어 디스켓 가격을 조금이라도 아낄 수 있다는 장점이 있다.

아카이브는 맨처음 PARC라는 이름으로 미국의 PKWARE SOFT에서 최초로 발표하였다. 그 뒤로 계속 버전업을 시키고 있는 중이며, 본지에서 소개하고자 하는 아카이브는 버전 3.5로 1987년도 판이다. 이것은 버전 1.0에 비해 압축속도가 25% 빨라졌으며, 복원속도도 50%가 빨라졌다. 그리고 −e, −x 옵션이 추가되면서 패스워드기능과 멀티태스킹기능 그리고 네트워크 기능이 확장되었다.

여기서 사용할 아카이브는 그 구성이 PKXARC.COM, PKARC.COM, PKXARCJR.COM, MAKESFX.COM로 구성되어 있고, 메모리의 총 용량은 165KB이다. 아카이브를 사용하려면 하드디스크나 최소한 1대 이상의 FDD를 갖춘 16비트의 IBM PC나 그 호환기종이 있어야 한다.

한가지 주의할 점은 아카이브를 사용하고자 한다면 항상 디스켓의 용량을 체크해 보길 바란다. 이것은 화일을 복원시킬 때 복원된 화일의 용량이 비어있는 용량보다 커서 용량이 모자라 에러가 발생할 수 있기 때문이다. 그러므로 복원시켜 저장할 드라이브에는 포맷된 새로운 디스켓을 넣도록 하는 것이 바람직하다.

이후의 설명은 보조기억장치로 하드디스크와 1대의 FDD가 준비된 상태에서 실행시키는 것을 토대로 하겠다. 먼저 디스켓을 A 드라이브에 넣고 'PKX35A35. EXE'라는 화일을 하드디스크에 저장한다. 여기서 기억할 것은 아카이브는 자신의 능력을 발휘하여 자신의 프로그램도 압축하여 가지고 있기 때문에 이 화일만 카피해서 실행시켜도 아카이브 유틸리티 사용에 필요한 모든 화일들이 자동으로 생성된다. 또한 이 유틸리티는 램상주상태에서 실행되므로 다른 프로그램 사용도중에 언제든지 불러낼 수 있다.

이제 드라이브 C상태에서

```
C:>PKX35A35
```

하면 실제로 데이터를 복원시키거나 압축시킬 때 사용되는 화일이 생성된다(화면 1 참조). 즉, PKARC.COM, PKXARC.COM, PKXARCJR.COM, MAKESFX.COM과 각 화일의 도큐먼트 화일이 생성된다. 이때 이것들은 길이가 길어 'Diskette full error'가 날 수 있으므로 이것들을 복원시켜 저장할 때는 가능하면 새 디스켓을 사용하기 바란다.

복원시킨 화일들의 기능을 대략 살펴보면 다음과 같다.

- PKARC.COM —— 화일을 압축시킬 때 사용되며 여러가지 옵션을 사용.
- PKXARC.COM —— 압축화일을 복원시킬 때 사용되며 여러가지 옵션을 사용.
- PKXARCJR.COM — 압축화일을 복원시킬 때 사용되며 여러가지 옵션 사용.
- MAKESFX.COM —EXE 확장자를 가진 압축화일을 만들어 PKXARC.COM 보다 더 쉽고 편리하게 복원하는 화일들을 만들 때 사용.

그러면 이러한 화일들의 기능에 관하여 좀더 자세히 알아 보자.

PKARC.COM

이 화일은 화일들을 압축하기 위해서 사용되며 사용구문은 다음과 같다.

```
C>PKARC [option] filename1,filename2 d:path
```

여기서 filenamel은 압축시킨 화일 이름이고, filename2는 압축시킬 화일의 이름이다.

예를 들어 현재의 디렉토리(여기서는 UTILITY 디렉토리내의 서브디렉토리인 ARCHIV)안에 있는 모든 화일들을 압축해서 archive란 화일이름을 주고 싶다면 다음과 같이 한다.

```
C:\UTILITY\ARCHIV>PKARC a archive
```

생성된 압축화일은 사용자 지정 화일이름에 .arc 확장자를 만든다. 여기에는 사용자에게 편리하도록 여러 옵션이 제공되고 있는 데 크게 복합옵션과 단순옵션으로 나눈다.

각 옵션에 대한 설명을 한 후 예를 들어 설명하겠다.

단순옵션

a : 일반적으로 압축시킬 때 쓰인다. 또한 이미 만들어져 있는 압축화일에 새로운 화일들을 압축시켜 더할 때 쓰인다.

(예) C:\UTILITY\ARCHIV>PKARV a:archive c:\ARCHIV *.doc c:\ARCHIV\ *.txt

C 드라이브에 있는 ARCHIV 디렉토리에서 ＊.doc와 ＊.txt 화일을 읽어서 압축시킨 후 A 드라이브에 archive란 화일이름으로 저장한다. 이때 컴퓨터는 같은 이름의 화일이 압축화일내에 저장되어 있다면 그 위에 오버라이트(overwrite)할 것인지를 사용자에게 묻는다.

d : 압축된 화일내에 있는 임의의 화일이름을 지울 때 사용된다.

(예) C:\UTILITY\ARCHIV>PKARC d archive *.doc

archive.arc 화일안에 존재하는 ＊.doc 화일을 지워버린다.

f : 압축화일 안에 들어 있는 각 화일들을 정리하고 다시 압축시킨다.

(예) C:\UTILITY\ARCHIV>PKARC f archive

archive.arc안에 들어 있는 화일을 정리하고 다시 압축한다.

m : 화일들을 압축시킨 후 지워버린다.

(예) C:\UTILITY\ARCHIV>PKARC m archive *.doc

＊.doc 확장자를 가지는 화일들을 읽어들인 뒤 archive.arc에 압축하여 저장한다. 그리고 압축한 화일은 지워버린다.

(주의 : 반드시 압축화일이 들어갈 디스켓의 용량을 확인하고 사용할 것. 실행중 메모리부족 에러가 나면 압축화일은 만들어지지 않고 화일만 지워진다.)

u : a와 같은 기능을 하며 압축화일 속에 같은 화일이름을 가진 화일이 이미 존재한다면 그 위에 오버라이트한다.

(예) C:\UTILITY\ARCHIV>PKARC u archive *.doc *.txt

＊.doc와 ＊.txt 화일을 현재의 디렉토리에서 선택한 후 archive.arc에 오버라이트한다. 즉 압축시킨 화일과 같은 이름의 화일이 이미 archive.arc에 존재한다면 가장 최근의 화일을 남겨둔다.

v : 압축화일 속에 있는 화일들에 대한 정보를 알려준다.

(예) C:\UTILITY\ARCHIV>PKARC v archive

archive안에 있는 화일들의 내용을 화면 2와 같이 보여준다. 여기서 length는 원래 화일의 길이이고, method는 압축시킨 방법이며, size는 압축후 총 길이를 의미하고, ratio는 원래 화일의 길이에서 압축후 생긴 공간의 비율이다. CRC는 Cyclic Redundancy Check로 이 값은 화일을 비교할 때 쓰이는데 만약 화일이름이 같아도 이 값이 같지 않으면 다른 화일로 처리한다.

l : PKWARE사의 소프트웨어 라이센스와 주소를 알려준다.

c : 화일에 주석문을 추가하거나 혹은 갱신할 때 사용한다.

(예) C:\UTILITY\ARCHIV>PKARC c archive

현재의 archive 화일안에 있는 화일들을 설명하는 주석문을 추가하거나 변경할 수 있게 해준다(화면 3 참조).

x : 압축화일 전체에 대한 주석문 추가

(c와 x, 이 두가지 옵션은 단독으로 쓰일 때는 반드시 압축화일이 존재해야 한다. 존재하지 않는 화일에 이 옵션을 쓰면 Can't use XXXX.ARC 혹은 Can't exit XXXX.ARC에러가 발생한다.)

g<password> : 이 옵션은 패스워드를 줄 수 있도록 되어 있다. 이 패스 워드는 압축화일을 복원시킬 때 일치하지 않으면 에러가 발생하며, 복원은 되나 이 화일을 실행시키면 컴퓨터가 에러를 일으킨다.

〔화면 1〕

```
PKSFX    FAST!    Self Extract Utility    Version 3.5    04-27-87
Copyright (c) 1987 PKWARE, Inc. All Rights Reserved. PKSFX/h for help

Unarcing: C:/UTILITY/ARCHIV/PKX35A35.EXE
unCrunching: MAKESFX.COM
unCrunching:PKARC.COM
unSquashing:PKARC.DOC
unCrunching:PKSFX.DOC
unCrunching:PKXARC.COM
unSquashing:PKXARC.DOC
unCrunching:PKXARCJR.COM
unCrunching:README.DOC
```

〔화면 2〕

```
PKARC    FAST!   Archive Create/Update Utility    Version 3.5    04-27-87
Copyright (c) 1986,1987 PKWARE Inc. All Rights Reserved.  PKARC/h for help

Searching Archive: ARCHIV.ARC

Filename       Length   Method    Size   Ratio    Date      Time    CRC
--------       ------   -------   ------ -----    ----      ----    ---
ARCHIV.BAK      17792  Crunched    9755   46%   01-01-80  00:46:36  9AB3
MAKESFX.COM      1358  Crunched     668   51%   04-27-87  00:00:00  98FE
PKARC.COM       19573  Crunched   16592   16%   04-27-87  00:00:00  168C
PKARC.DOC       27270  Squashed   12100   56%   04-27-87  00:00:00  E419
PKSFX.DOC        1534  Crunched     922   40%   04-27-87  00:00:00  C4E5
PKX35A35.EXE    71680  Squeezed   70783    2%   01-01-80  00:20:54  9DBA
PKXARC.COM      12242  Crunched   10627   14%   04-27-87  00:00:00  5F64
PKXARC.DOC      18717  Squashed    8793   54%   04-27-87  00:00:00  FFF0
PKXARCJR.COM    12110  Crunched   10542   13%   04-27-87  00:00:00  ADFE
README.DOC        540  Crunched     389   28%   04-27-87  00:00:00  C4A1
----           ------             ------ -----
0010           182816             141171  23%
```

〔화면 3〕

```
PKARC    FAST!   Archive Create/Update Utility    Version 3.5    04-27-87
Copyright (c) 1986,1987 PKWARE Inc. All Rights Reserved.  PKARC/h for help

Updating Archive: ARCHIV.ARC
ARCHIV.BAK
Old Comment : conffresing documemt file
New Comment ? document file
MAKESFX.COM
Old Comment :
New Comment ?
PKARC.COM
Old Comment :
New Comment ?
```

(예) C>PKARC agsecret archive *.doc

('ag'는 옵션, 'secret'는 패스워드)

✱.doc 화일을 읽어서 secret란 암호를 가진 archive.arc 화일에 추가한다. 여기서 주의할 것은 g옵션은 화일의 압축과는 무관한 옵션이므로 반드시 다른 압축옵션과 같이 쓰여야 한다. 그리고 g와 패스워드 사이에 공백이 있으면 안된다.

복합 옵션

PKARC는 두가지의 옵션을 동시에 수행하는 것과 같은 기능이 있다. 이것은 압축화일에 주석문을 넣을 수 있다는 것으로 이들 옵션은 c나 x옵션에 종속된 옵션들이기는 하지만 화일을 정리하여 압축시켰을 때 화일이름만으로 압축된 프로그램의 성격을 알 수가 없을 경우, 비슷한 것끼리 정리해 압축시켜 놓지 않아도 주석문을 달아 놓으면 편리할 때가 있다.

vc : 화일 주석문을 열거한다.

(예) C:\UTILITY\ARCHIV>PKARC vc archive

현재의 디렉토리안에 있는 archive 압축화일들의 내용을 주석문과 함께 보여 준다.

주석문은 30자까지 가능하다.

ax, ux, fx, mx : x 앞에 있는 방법대로 압축화일을 만든 후 압축화일에 대한 주석문을 받을 수 있게 프롬프트가 나타난다.

(예) C:\UTILITY\ARCHIV>PKARC ax archive *.dat

현재의 디렉토리에서 ✱.dat화일을 선택해서 archive.arc 화일에 추가시키고 각 화일들의 주석문을 받을 수 있는 상태로 되면서 프롬프트가 나타난다.

ac, uc, fc, mc : c 앞에 있는 방법대로 압축화일을 만든 압축된 화일들에 대한 주석문을 받을 수 있게 프롬프트가 나타난다.

(예) C:\UTILITY\ARCHIV>PKARC ac archive *.dat

현재의 디렉토리에서 ✱.dat화일을 선택해서 archive.arc 화일에 추가시키고 각 화일 주석문을 받을 수 있는 상태로 되면서 프롬프트가 나타난다.

axc : x 옵션과 c옵션의 기능을 한꺼번에 실행시켜 화일을 추가시킬 때 사용할 수 있다.

(예) C:\UTILITY\ARCHIV>PKARC axc archive

현재의 디렉토리에서 화일을 선택해서 archive.arc화일에 추가시키고 화면 4와 같이 .arc화일과 각 화일들의 주석문을 받을 수 있도록 프롬프트가 나타난다.

(예) C:\UTILITY\ARCHIV>PKARC cx archive

현재의 archive 압축 화일안에 있는 화일들 각각의 주석문과 archive.arc 화일의 전체 주석문을 추가하거나 변

〔화면 4〕

```
PKARC    FAST!   Archive Create/Update Utility    Version 3.5    04-27-87
Copyright (c) 1986,1987 PKWARE Inc. All Rights Reserved.  PKARC/h for help

Updating Archive: ARCHIV.ARC
Updating: ARCHIV.BAK    analyzing, (46%) crunching, done.
Updating: MAKESFX.COM   analyzing, (51%) crunching, done.
Updating: PKARC.COM     analyzing, (16%) crunching, done.
Comment ?
```

경할 수 있게 해준다.

이것들 이외에도 여러가지 옵션들을 이용하여 사용자가 더욱 편리하게 쓸 수 있도록 하는 기능이 있으므로 각자 필요한 옵션들을 조합하여 써보기 바란다. 그리고 이 PKARC를 사용할 때 옵션을 사용하지 않을 경우, 뒤에 압축화일 이름이 옵션과 동일한 철자로 시작하면 옵션으로 간주해 버린다. 이럴 경우 사용자와 전혀 다른 의도로 사용되므로 가능하면 옵션을 사용하기 바란다.

다음은 PKARC.COM을 사용하면서 자주 나타나는 에러메시지와 그에 대한 설명이므로 사용상의 참고가 되길 바란다.

PKARC : Unknown option

명령어에 쓰인 옵션이 잘못 쓰였음.

PKARC : Insufficient Memory

PKARC를 실행시키기에는 메모리가 부족하다.

PKARC : No files specified for deletion.

삭제 화일이름이 현재의 .arc 화일에 존재하지 않음. *.*은 d 옵션에서는 쓰일 수 없음.

PKARC : Can't find XXXX.YYY

XXXX.YYY 화일을 찾을 수 없음.

PKARC : No file(s) found.

추가 혹은 갱신할 화일을 찾을 수 없음.

PKARC : Can't creat file.

출력화일이 존재하지 않을 때 나타나는 에러이다.

PKXARC.COM

이제까지는 화일을 압축시킬 수 있도록 하였다. 이제는 압축시킨 화일들을 원래의 상태로 복원시켜 보자. 압축화일을 복원시킬 때 사용되는 구문은 다음과 같다.

```
C>PKXARC [option] filename d:path
```

예를 들어 만약 C 드라이브에 있는 archive란 압축화일을 A 드라이브에 복원시킨다면

```
C:\UTILITY\ARCHIV>PKXARC archive a:
```

하면 된다. 그러면 화일에 복원되는 화일들의 이름이 하나씩 나타나며 드라이브 A에 저장된다.

여기서 한가지 기억할 것은 PKXARCJR. COM 화일은 PKXARC.COM과 유사한 기능을 가지고 있다는 것이다. 그러므로 여기서 PKXARCJR.COM에 대한 설명은 생략한다.

PKXARC.COM에서도 여러가지 옵션들을 사용할 수 있다. 단, 여기서 옵션을 사용할 때는 반드시 앞에 "—" 부호를 붙여야 한다.

그러면 다음에서 여러가지 옵션에 관하여 예와 함께 살펴보자.

—r : 복원된 화일들과 같은 이름의 화일이 존재할 때 그 위에 덮어 씌운다.

실제로 복원시킨 화일과 같은 이름의 화일이 이미 사용 드라이브안에 존재한다면 컴퓨터는 오버라이트할 것인지를 사용자에게 묻는다. 이것은 때때로 귀찮은 작업이 된다. 따라서 —r 옵션을 사용한다면 이런 불편함을 제거할 수 있다.

다음의 예를 살펴보자.

```
C:\UTILITY\ARCHIV>PKXARC a:* b:
```

드라이브 A에 있는 모든 압축된 화일들을 복원시켜 드라이브 B에 저장한다.

```
C:\UTILITY\ARCHIV>PKXARC archive c:
```

현재 드라이브에서 archive.ARC 화일을 찾아 복원시키면서 복원시킨 화일을 드라이브 C에 저장한다.

```
C>PKXARC c:\dbase\TAX *.WKS
```

C 드라이브에 있는 디베이스 디렉토리내의 TAX.ARC 압축화일에서 .WKS 확장자를 가지는 화일들을 복원시켜 같은 드라이브의 서브 디렉토리에 저장한다.

위의 3가지 경우는 압축화일에서 화일을 복원시켜 저장하면서 같은 이름의 화일이 사용드라이브(혹은 디렉토리)안에 존재하면 이미 존재하는 화일위에 다시 복원시킨 화일을 덮어 씌울 것인지 다음과 같은 메시지를 나타난다.

```
Warning! File XXXXXX.YYY already exists!
overwrite(y/n)?
```

그러나 이것은 가끔 귀찮을 때가 많다. 그러므로 앞에서 설명한 "—r" 옵션을 쓰면 무조건 오버라이트 해버린다.

```
C>PKXARC -r a:\ C:\UTILITY\ARCHIV\*.doc
 *.txt
```

C 드라이브의 ARCHIV 디렉토리 안에 있는 모든 압축화일에서 .doc, .txt 확장자를 가진 화일만 복원시켜 A 드라이브의 루트디렉토리에 무조건 저장한다.

—c : 압축된 화일들의 도큐먼트화일을 실행시켜 내용을 화면에 보여 주지만 복

원시키지는 않는다.

(예) C:\UTILITY\ARCHIV>PKXARC -c *

ARCHIV 디렉토리안에 있는 압축화일의 모든 내용이 나타난다. 프로그램이 압축되어 있을 경우 프로그램 리스트가, 도큐먼트 화일이 있을 때에는 도큐먼트 화일이 실행되며 내용이 나타난다.

–p : 압축된 화일들의 내용, 또는 도큐먼트 화일을 실행시켜 그 내용을 프린터에 출력하지만 내용을 복원시키지 않는다.

–t : 압축화일 속에 있는 화일들을 테스트한다.

즉, 각 압축화일들의 CRC 값을 계산하여 저장된 CRC 값과 비교한다. (컴퓨터는 CRC 값과 화일 이름이 일치하지 않을 경우는 CRC check 에러가 발생한다.)

(예) C:\UTILITY\ARCHIV>PKXARC -t *

현재의 디렉토리안에 있는 압축화일의 모든 CRC 값을 체크한다(화면 5 참조).

–v : 압축화일 속에 있는 화일이름, 압축되기 전 화일의 길이, 압축방법, 압축 후 길이, 압축비율, CRC값, 화일 생성일, 화일 생성시간 등의 화일들에 관한 정보를 스크린에 나타낸다.

(예) C:\UTILITY\ARCHIV>PKXARC -v * archive

ARCHIV 디렉토리안에 있는 archive 압축화일의 모든 내용을 보여준다.

–l : PKWARE사의 소프트웨어 라이센스와 주소를 알려준다.

–e, –x : 압축된 화일을 복원시킨다.

(예) C:\UTILITY\ARCHIV>PKXARC -e *

ARCHIV 디렉토리안에 있는 모든 압축화일을 찾아서 복원시킨다(화면 6 참조).

–g〈password〉: 패스워드를 가진 압축화일을 복원시킬 때 사용한다.

이것을 사용하지 않고 압축화일을 복원시키게 되면 경고 메시지가 나타나고 패스워드를 가진채로 복원되어 나온다. 여기서 주의할 것은 이 화일을 사용하면 시스템이 다운된다는 것이다.

다음은 PKXARC.COM을 사용할 때 나타나는 에러 메시지이다.

```
Warning! File XXXXXX.YYY already exists!
overwrite (y/n) ?
```

압축된 화일로부터 복원되어 나오는 화일이 이미 작동 디스크상에 존재할 때 그 위에 덮어 씌우는 지의 여부를 묻는다. "–r" 옵션을 쓰면 이 에러는 발생하지 않는다.

```
Unknown Packing method for : XXXXXX.YYY
```

〔화면 5〕

```
PKXARC    FAST!   Archive Extract Utility    Version 3.5    04-27-87
Copyright (c) 1986,1987 PKWARE Inc. All Rights Reserved. PKXARC/h for help

Searching: ARCHIV.ARC
Testing: ARCHIV.BAK    Ok
Testing: MAKESFX.COM   Ok
Testing: PKARC.COM     Ok
Testing: PKARC.DOC     Ok
Testing: PKSFX.DOC     Ok
Testing: PKX35A35.EXE  Ok
Testing: PKXARC.COM    Ok
Testing: PKXARC.DOC    Ok
Testing: PKXARCJR.COM  Ok
Testing: README.DOC    Ok
```

〔화면 6〕

```
PKXARC    FAST!   Archive Extract Utility    Version 3.5    04-27-87
Copyright (c) 1986,1987 PKWARE Inc. All Rights Reserved. PKXARC/h for help

Unarcing: ARCHIV.ARC
unCrunching: ARCHIV.BAK
Warning!  file ARCHIV.BAK already exists!  overwrite (y/n)? n
ARCHIV.BAK not extracted.
unCrunching: MAKESFX.COM
Warning!  file MAKESFX.COM already exists!  overwrite (y/n)? n
MAKESFX.COM not extracted.
unCrunching: PKARC.COM
Warning!  file PKARC.COM already exists!  overwrite (y/n)? n
PKARC.COM not extracted.
unSquashing: PKARC.DOC
Warning!  file PKARC.DOC already exists!  overwrite (y/n)? n
PKARC.DOC not extracted.
unCrunching: PKSFX.DOC
Warning!  file PKSFX.DOC already exists!  overwrite (y/n)? n
PKSFX.DOC not extracted.
unSqueezing: PKX35A35.EXE
Warning!  file PKX35A35.EXE already exists!  overwrite (y/n)?
```

압축화일이 알 수 없는 방법으로 압축되어 있을 때 나타난다.

```
Warning! File XXXXXX.YYY fails CRC check
```

CRC 계산값이 저장값과 일치되지 않을 때 발생한다.

```
XXXXXX.YYY is not an archive
```

XXXXXX.YYY 화일이 압축화일 속에 존재하지 않는다.

```
PKARC : Can't find XXXXXX.YYY
```

주어진 .arc 화일의 이름을 찾을 수 없다.

```
PKXARC : Can't create XXXXXX.YYY
```

XXXXXX.YYY 화일을 만들 수 없다. 출력 디렉토리가 존재하지 않거나 공간이 없음을 의미한다.

```
PKXARC : File has invalid decode tree
```

디코드하는 화일이 존재하지 않음.

```
PKXARC : Disk full
```

작동 디스크가 꽉찼음.

MAKESFX.COM

MAKESFX.COM 화일은 PKSFX.PGM 화일을 생성하는데 쓰인다. 이 화일을 작동시키기 위해서는 같은 디렉토리안에 PKX35A35.EXE 화일이 반드시 존재해야 한다.

```
C:\UTILITY\ARCHIV>MAKESFX
```

하면 PKSFX.PGM 화일이 생성된다.

PKSFX.COM

이것은 바로 압축화일로부터 PKXARC를 사용하지 않고 화일들을 복원시킬 수 있는 .EXE 화일을 생성하는데 쓰인다.

이 유틸리티를 처음 실행시킬 때

```
C:\UTILITY\ARCHIV>PKX35A35 ↵
```

를 실행하면 화일들이 바로 복원된다.

.EXE 화일을 생성시키는 방법은 다음과 같다.

1. 원하는 화일들을 PKARC를 사용하여 축약시킨다.
2. C>MAKESFX ↵를 하면 PKSFX.PGM 화일이 생성된다.(이때 반드시 PKX35A35.EXE 화일이 존재해야 한다.)
3. PKSFX.PGM 화일 끝에 다음과 같은 방법으로 1번의 방법으로 생성된 화일을 복사한다.

```
C>COPY PKSFX.PGM+archive.arc archive.exe ↵
```

archive.arc는 1에서 생성된 압축화일의 이름이다. 이제 archive.exe라는 자체적으로 복원시킬 수 있는 화일이 생성된다.

이제 C>archive↵를 하면 압축되었던 화일들이 자유를 찾은 것처럼 마구 쏟아져 나온다.

맺음말

지금까지 설명한 아카이브를 통하여 우리는 화일을 압축시킬 수 있어 시간과 메모리를 절약할 수 있게 되었다. 또한 아카이브는 주석문을 사용할 수 있는 옵션이 제공되어 사용자가 디스켓을 정리하는 데도 도움을 준다. 그리고 이 아카이브의 기능과 유사한 기능을 가진 N Archive라는 유틸리티는 POP ON 메뉴형식을 제공하여 사용을 더욱 편리하게 하고 있다.

그러나 필자의 경험에 미루어 볼 때 이제까지 소개한 아카이브는 압축화일을 복원시키는 옵션부분에서 복합옵션을 사용할 수 없다는 문제점이 있다. 그러므로 이러한 문제점을 해결시키면서, 압축방식을 개선하여 압축시킬 수 있는 비율을 증가시킨다면 매우 유용한 유틸리티가 될 것이라 기대한다. PC

배치화일로 인스톨 작업을

컴퓨터 사용자들은 모든 소프트웨어를 디스켓을 통하여 얻은 후 동일한 백업디스켓을 만들어 사용하게 된다. 이같은 이유는 예기치 않은 불상사를 대비하여 원본디스켓을 안전한 장소에 보관하기 위함이다.

근래에 들어서는 하드디스크가 많이 보급되면서 각종 컴파일러나 유틸리티 프로그램, 패키지등 중요한 시스템 프로그램들은 하드디스크의 적당한 디렉토리에 보관하고 여기서 직접 실행시키는 경우가 많아지고 있다.

실제로 사용자들이 임의의 프로그램 디스켓을 처음 구하고 디렉토리를 살펴보면 대부분 인스톨 프로그램을 볼 수 있을 것이다. 일반적으로 인스톨 프로그램이란 하드디스크에서 실행할 수 있도록 적당한 서브디렉토리들을 만들고 디스켓의 화일들을 해당 디렉토리에 복사한 다음 실행에 적합한 환경을 만들어 놓는 화일을 말하는 것으로 하드디스크를 전제로 하는 자동 실행화일을 말하는게 보통이다. 경우에 따라서는 디바이스 드라이버 등을 포함하는 CONFIG.SYS를 인스톨 프로그램에 첨가하여 실행환경을 만들때도 있지만 여기서는 원본디스켓과 똑같은 디렉토리 구조를 하드디스크에 자동적으로 설치하는 배치화일인 경우에 한하여 알아보자.

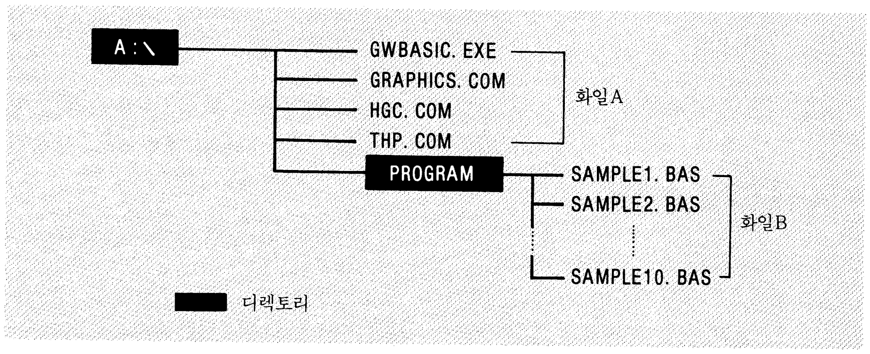
여기서 설명하는 배치화일은 누구나 알고 있으면서도 지나치기 쉬운 인스톨 작업에 대한 경우로써 초보자들에게는 디렉토리 개념을 이해하는데도 도움이 되리라 본다.

플로피디스크와 하드디스크가 보조

〔보기 1〕 INSTALL1.BAT

```
A>COPY CON INSTALL1.BAT
CLS
REM --- Directory name is BASIC1 ---   ← 화면메시지
C:             ┐
MKDIR BASIC1   ┘ 하드디스크로 제어를 옮긴 후 BASIC1 디렉토리를 만든다.
CD BASIC1      ┐
COPY A:*.*     ┘ BASIC1 속으로 제어를 옮긴 후 그곳에 A의 모든 화일을 복사한다.
CD\
A:
^Z
```

〔그림 1〕 보기 2의 디렉토리 구조



〔보기 2〕 INSTALL2.BAT

```
A>COPY CON INSTALL2.BAT
CLS
REM --- RootDirectory name : BASIC1
REM --- SubDirectory name : PROGRAM
C:
MKDIR BASIC1   ┐
CD BASIC1      │ BASIC1 디렉토리를 만들고 그림 1의 화일 A를 복사
COPY A:*.*     ┘
MKDIR PROGRAM                  ┐ BASIC1 속에 다시 PROGRAM 서브디렉토리를
COPY A:PROGRAM\*.* PROGRAM     ┘ 만들고 해당화일 복사
CD\
A:
^Z
```

기억매체라는 것은 누구나 다 아는 사실이다. 특히 하드디스크는 용량이 방대한 기억매체이므로 여러 종류의 프로그램들을 아무 구별없이 한 장소에 보관한다면 특정 화일을 찾는데도 진땀을 흘려야 할 것이다. 왜냐하면 수백개의 화일들이 한장소에 밀집해 있기 때문이다. 이런 문제를 해결하기 위해 하드디스크를 여러 영역으로 분리하여 각 영역마다 같은 종류의 화일들로 저장시키고 있다. 이렇게 디스크를 여러개의 방으로 나눈 것을 디렉토리(Directory) 라고 부르며 DOS의 MKDIR(Make Directory)명령으로 사용자들이 직접 만들 수 있다.

하드디스크 또는 플로피디스크 전체의 공간을 루트디렉토리라고 하고 그 안에 만들어진 디렉토리를 서브디렉토리라고 부르는데 서브디렉토리 안에서 다시 방들을 만들 수도 있다. 이는 마치 큰 건물안에 A, B 두 회사 사무실이 있는데 B회사 사무실 안으로 들어가면 다시 영업부, 경리부, 기획부 등의 사무실로 나누어 있는 이치와 흡사하며 이들 사무실 즉 디렉토리에 들어가고 나오기 위해서는 DOS의 CD(Change Directory) 명령을 이용한다.

가장 간단한 경우로서, A드라이브에 있는 디스켓의 내용을, 하드디스크에 적당한 방을 만들어 저장시키는 배치화일을 만들어 보자. 베이직 인터프리터와 베이직 프로그램 화일들이 들어있는 디스켓의 내용을 하드디스크로 옮기고 싶다면 보기 1과 같은 배치화일을 만들어 디스켓 A에 저장시킨 후 A 드라이브 상태에서

```
A>INSTALL1
```

을 입력하면 모든 작업이 이루어지게 된다. 여기서는 디렉토리 이름을 BASIC1으로 했지만 이것은 사용자가 임의로 정할 수 있으며 배치화일 이름 또한 임의로 정할 수 있다.

이번에는 루트디렉토리에는 인터프리터 화일이 들어 있고 PROGRAM이란 서브디렉토리에는 프로그램 화일들이 들어있는 디스켓 A의 내용(그림 1)을 하드디스크로 옮겨 보자. 이 경우도 배치화일에서 BASIC1 디렉토리 속에 다시 PROGRAM이라는 서브디렉토리를 만들어 주면 된다(보기 2 참조).

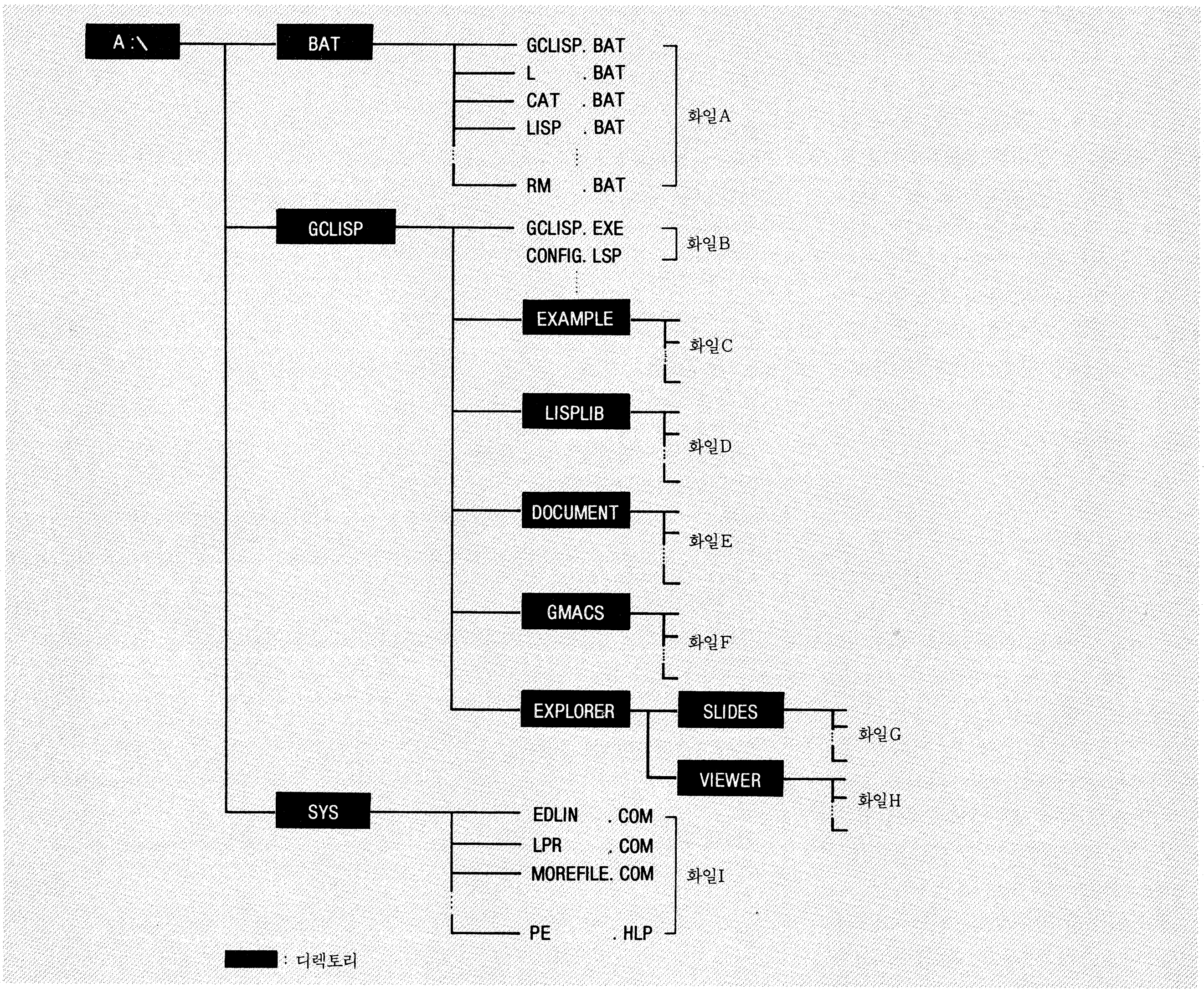
마지막으로 보기 3은 실예로써 GCLISP 디스켓(1.2MB) 2장을 하드디스크로 설치하는 인스톨 프로그램을 보인 것으로 배치화일의 중간에서 'ECHO'와 'PAUSE'명령으로 2번째 디스켓을 A드라이브에 넣으라는 메시지도 포함하고 있다.

그림 2는 원본 디스켓의 디렉토리 구조를 나타낸 것이다.

〔보기 3〕 INSTALL3.BAT

```
A>COPY CON INSTALL3.BAT
CLS
REM --- Installing ...
C:
MKDIR BAT
COPY A:BAT\*.* BAT  ←—— 화일A
MKDIR GCLISP
CD GCLISP
COPY A:*.*  ←—— 화일B
MKDIR EXAMPLE
MKDIR LISPLIB
MKDIR DOCUMENT
MKDIR GMACS
COPY A:EXAMPLE\*.* EXAMPLE      ┐
COPY A:LISPLIB\*.* LISPLIB      │ 화일C, D, E, F
COPY A:DOCUMENT\*.* DOCUMENT    ┘
COPY A:GMACS\*.* GMACS
REM --- Insert the Second Diskette ...  ┐ 두번째 디스켓을 넣고
PAUSE                                   ┘ 아무키나 누르면 계속 실행된다.
MKDIR EXPLORER
CD EXPLORER
MKDIR SLIDES
MKDIR VIEWER
CD A:\GCLISP\EXPLORER
COPY A:SLIDES\*.* SLIDES  ┐
COPY A:VIEWER\*.* VIEWER  ┘ 화일G, H
CD A:\SYS
CD\
MKDIR SYS
COPY A:*.* SYS  ←—— 화일I
A:
CD A:\
CD C:\
REM --- Install is DONE ---
^Z
```

〔그림 2〕 보기 3의 디렉토리 구조



전자·통신·컴퓨터분야의 명문

도서출판 Ohm사의 도서안내

새로운 기술정보를 제공하는 Ohm사는
공학도들의 등불이 되고자 노력하고 있습니다.

공학도서

디지탈 신호처리
황병원 저/신국판/338면/6,500원

디지탈 공학
김태중·최희태·허정인 공저/신국판/222면/5,500원

마이크로파 공학
김정기외 3명/신국판/376면/7,000원

전자회로
김철수외 2명/신국판/478면/6,500원

광파이버의 기초
전자과학 편집부편/신국판/270면/4,000원

엘렉트로닉스 공작기술
전자과학 편집부편/신국판/278면/4,000원

제어공학 시리즈
- 제1권 순서제어 황병원 저/신국판/484면/7,000원
- 제2권 전자제어 황병원 저/신국판/378면/5,500원
- 제3권 디지탈제어 황병원 저/신국판/349면/5,000원
- 제4권 마이컴제어(기초) 황병원 저/신국판/272면/4,000원
- 제5권 마이컴제어(응용) 황병원 저/신국판/298면/4,500원

이공학계 총서 시리즈
- 제1권 푸리에해석 황병원 편저/신국판/380면/6,000원
- 제2권 벡터해석 황병원 편저/신국판/438면/7,000원
- 제3권 확률 및 통계 황병원·이기수 공저/신국판/450면/7,000원
- 제4권 자동제어 황병원·이용두 공저/신국판/456면/7,000원
- 제5권 반도체공학 황병원 편저/신국판/근간예정
- 제6권 펄스공학 김원후·황병원 공저/신국판/370면/6,000원

컴퓨터 도서

근거리통신망(LAN)
정진욱·변옥환 공저/4·6배판/446면/8,500원

데이터통신과 컴퓨터 네트워크
정진욱·변옥환 공저/4·6배판/670면/9,500원

8BIT MICROPROCESSOR
전자과학 편집부편/4·6배판/342면/5,200원

16BIT MICROPROCESSOR
전자과학 편집부편/4·6배판/337면/5,200원

신규 FORTRAN의 기초
황병원 편저/4·6배판/266면/5,500원

소프트웨어와 프로그래밍
정진욱 저/신국판/257면/4,800원

APPLE Ⅱ BASIC GRAPHICS
박치욱 편저/신국판/195면/3,500원

과년도 기사문제집 시리즈

유선설비기사 1·2급
Ohm사 편집부편/신국판/304면/3,500원

무선설비기사 1·2급
Ohm사 편집부편/신국판/264면/3,500원

사전류

신통신기술용어해설집
윤승은·황영국 공저/국판/674면/8,000원

최신 전자용어사전
전자과학 편집부편/국판/620면/7,000원

마이크로컴퓨터 용어사전
Ohm사 편집부편/국판/416면/6,000원

圖書出版 **Ohm社**
㊃140-150 서울특별시 용산구 갈월동 17 세기빌딩
TEL : 754-1044, 2902, 1751

●PC레이다●

국내소식

–체신부
장비의 표준화 추진

체신부는 정보통신 기술기준 기본개편 방안을 마련하고 연내에 체신부, 전기통신공사, 데이타통신, 전자통신연구소 및 통신관련 생산업체와 공동으로 전담반을 구성해 기술기준체계와 장비의 표준화를 마련할 계획이다.

대상은 ISDN(종합정보통신망)의 실용화에 대비하여 이기종 컴퓨터의 각종 정보통신기기와의 호환성과 통신단말시설 관련기술 등이다. 체신부는 이와 함께 기술기준의 개선 보완을 위해 기본기술관련규정은 3년마다, 세부적 기술관련규정은 매년 개선하여 기술변화 추세에 맞춰 나가기로 했다.

–과기처
"정보산업발전 5개년계획" 발표

정부는 미래 핵심산업으로 부상할 정보산업을 육성·발전시키기 위해 범부처적으로 "정보산업발전 5개년계획"을 수립하고 적극 추진할 방침이다.

이상의 과기처장관은 한국 정보산업연합회 주최로 열린 과기처장관초청 오찬간담회에서 "정보산업을 육성·발전시키는 것이 곧 정보화사회로 조기진입을 실현하는것"이라고 전제하고 이를 위해 과거 경제개발 5개년계획과 비슷한 형태의 정보산업발전 5개년계획을 수립, 적극추진할 것이라고 밝혔다.

이장관은 현재의 정보 관련산업법령으로는 정보화 추세를 수용하기 어렵다고 지적하고 정보화사회 구축을 제도적으로 뒷받침할 수 있는 "정보화사회추진법(가칭)"을 제정하겠다고 밝혔다. 그리고 소프트웨어 등 정보산업관련 지적소유권 보호업무를 전담하는 정보청을 설립하여 선진국과의 통상마찰을 해소할 수 있도록 하겠다고 하였다.

이장관은 이밖에 현재 15개로 결성되어 있는 정보산업분야의 연구조합에 대한 세제·금융상의 지원조치를 확대하여 연구개발을 활성화시키고 밴처캐피탈자금이 정보산업에 집중 투입되도록 하겠다고 밝혔다.

–통신개발연구원
국가정보화 측정지표 등 개발키로

통신개발연구원(원장 : 김세원)은 올해 국가정보화 측정지표개발, 전기통신요금 규제정책에 관한 연구 등 정보화사회의 조기정착을 위한 연구분야에 중점을 두어 개발하기로 했다.

통신개발연구원은 올해 사업방향을 마련, 중점 연구사업으로 우선 사회정책 분야에서 국가정보화 측정지표 개발에 관한 연구와 정보화 사회에 대비한 프라이버시 보호대책에 관한 연구 등을 추진할 계획이다.

또 요금 및 수요정책분야에서는 전기통신요금 규제정책에 관한 연구, 정보산업 실태조사와 정보통신서비스 수요분야 등을 다루기로 했다.

통신개발연구원은 이같은 사업을 추진하기 위해 박사급 10명을 포함한 20명의 연구인력을 보강할 계획이다.

–특허청
한글검색시스템 개발키로

특허청은 금년에 특허업무의 전산화계획으로 분류기호를 한글화하여 심사, 검색 등에 이용할 수 있는 한글검색시스템을 개발하기로 했다.

개발될 시스템은 국제특허분류(IPC)기호에 따라 검색하는 작업을 발명 명칭에 따라 쉽게 할 수 있도록 하는 시스템이다.

이와 병행하여 특허청은 90년부터 새로 개정되는 IPC기호들을 89년 4월쯤 입수하여 한글색인 발간을 추진하기로 했다.

동 시스템이 완성되면 심사관의 심사업무, 신속성 및 효율성 등의 효과를 기대할 수 있고 한글로 된 발명명칭을 이용하여 발명에 관한 사항을 컴퓨터를 통해 알 수 있게 된다.

–갑을방적
테레비디오 지분 51% 인수키로

갑을방적(대표 : 박창호)은 퍼스컴전문 생산업체인 테레비디오컴퓨터를 인수하기로 했다.

테레비디오는 지난해 노사분규의 후유증과 본사인 테레비디오시스템사의 경연난 등으로 자본금의 51%인 52억

5천만원의 지분을 회사경영권과 함께 갑을방적으로 매각하기로 했다.

한편 갑을방적은 지분인수가 끝나는 올 하반기부터 직접경영체제에 돌입하여 오는 90년대 초반 테레비디오컴퓨터의 주식을 공개할 것으로 알려졌으며, 주요 생산제품인 16비트, 32비트 퍼스컴의 부가가치를 높이기 위한 전략으로 소프트웨어 및 주변기기 개발에도 적극 투자할 것으로 보인다.

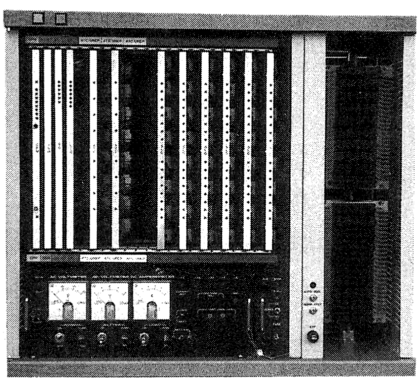
테레비디오컴퓨터는 87년 5억9천만원의 적자를 기록했고 지난해도 노조파업 등으로 9억5천만원의 적자를 기록함으로써 2년 연속 불황에 허덕여 왔다.

테레비디오컴퓨터는 지난 81년 미 테레비디오시스템사의 국내지사로 출발하여 83년 현지법인 설립과 함께 연간 50만대의 퍼스컴 및 CRT터미널을 생산할 수 있는 공장(연건평 4천20평)을 건립하였고 지난해 미 본사에 5천만달러의 OEM수출을 기록한바 있다. 현재 자본금은 1백2억9천만원으로 부채비율은 40%이하로 매우 양호한 편이다.

-금성반도체

사설교환기 4기종 체신부 형식승인 취득

금성반도체(대표 : 이만용)는 자체개발한 스타렉스시리즈인 STAREX-100D, STAREX-200HD, STAREX-I-MS 등 3종과 NEAX-2400IMS 등 모두 4종을 국내에서는 처음으로 체신부로부터 형식승인 제1호부터 4호까지 취득했다.



이로써 동사는 자체검사만으로 사설교환기를 시장에 내놓을 수 있게되어 제품의 출하시간이 단축되고 수용자에게는 제품을 보다 적시에 공급할 수 있게 되었다.

형식승인제도란 다른 공산품의 "품"자 제도와 비슷한 것으로 체신부는 이용자의 안정성 및 편익증진과 국내통신망의 신뢰성 및 호환성의 확보 등을 위해 품질이 우수하고 신뢰성이 뛰어난 전기통신제품에 대하여 이 제도를 시행해 왔다.

종래의 사설교환기업체가 제품을 수요자에게 공급하려면 반드시 한국전기통신공사(KTA) 품질보증단으로부터 2주간에 걸친 성능시험 검사후 테스트 결과를 발급받아야 했다.

금성반도체가 형식승인을 취득한 이들 사설 교환기는 첨단 디지탈교환기로 이미 수출되고 있고 국제전신전화자문위원회(CCITT)의 표준권고사항을 채택, 다른 공중통신망과의 상호접속도 용이하다.

금성반도체는 이번 체신부의 형식승인 취득을 계기로 올해 사설교환기의 매출목표를 작년대비 30%정도 올려잡는 동시에 ISDN(Integrated Service Digital Network : 종합정보통신망)에 대비한 PBX의 개발도 계획하고 있는 것으로 알려졌다.

-금성기전

서보모터 OEM 수출키로

금성기전(대표 : 홍종선)은 국내 처음으로 로보트, NC공작기계 등 자동화기기의 정밀구동형 핵심부품인 서보모터 및 모터드라이버의 수출에 나선다.

일본의 다이또(대동)제작소와 OEM 계약을 체결하여 앞으로 5년에 걸쳐 매년 DC서보모터 3만6천대 모터드라이버 6천대를 각각 수출할 예정이다.

DC서보모터 및 모터드라이버는 지난해 7월 자체연구소에 의해 개발에 성공한 DC 75V급 200W와 300W의 2기종으로 산업용로보트 및 NC공작기계 등 각종 자동화기기에 두루 사용할 수 있는 제품이다.

지금까지 서보모터는 국내 기술개발 부족으로 대부분 일본으로부터 수입에 의존해 왔다.

금성기전은 이번 수출로 국산 서보모터에 대한 인식이 크게 제고될 것으로 기대하고, 수입대체 및 수출확대에 적극 나설 방침이다.

-금성소프트웨어

고부가가치 패키지에 주력키로

소프트웨어산업이 수출 유망 성장산업으로 부각됨에 따라 국내 소프트웨어업체들의 수출방식이 단순용역에서 고부가가치위주의 혁신적인 수출방식으로 바뀌어 가고 있다.

관련업계에 따르면 국내 소프트웨어 업체들은 그동안 기술축적이 어렵고 실이익이 별로 없은 프로그래머의 단순용역 수출방식을 탈피하고, 고부가가치의 특정 소프트웨어패키지 수출이나 국제통신회선을 활용하여 시스템설계 및 프로그래밍 등 소프트웨어개발 전공정에 공동참여하는 획기적인 방식으로 전환하고 있다.

이같은 현상이 일어나는 원인으로는 소프트웨어 패키지에 의한 부가가치가 단순용역 수출보다 휠씬높고, 통신기술의 발달로 컴퓨터간 네트워크가 더욱 쉬워졌으며, 시스템설계 등 소프트웨어 엔지니어링 기술이 점차 고도화되고 있기 때문으로 해석된다.

이러한 수출방식이 널리 채택되자 금성소프트웨어 (대표 : 황칠봉) 는 해외시장 확보전략의 일환으로 지난해 7월 미국 아티스사와 소프트웨어 기술제휴를 포함한 수출계약을 체결, 해외유통망을 통한 패키지수출에 적극 나서고 있다.

–대우통신
컬러그래픽 터미널 양산

대우통신(대표 : 박성규)은 최근 DEC의 TV-340과 호환가능한 14인치 컬러 그래픽터미널 DT-9420을 개발, 양산에 들어 갔다.

이 제품은 멀티프로세서 아키텍처를 채택, 그래픽 에뮬레이션 및 VT-340의 문자처리 기능을 갖추었으며 서로 다른 호스트 컴퓨터로부터 데이터를 받아 동시에 처리할 수 있는 화면동시 분할기능을 구비한 것이 특징이다. 최대 데이터전송속도는 38.4kbps.

대우통신은 DT-9420개발을 통해 축적된 기술로 시장경쟁력 확보를 위해 DEC VT330과 호환가능한 14인치 컬러그래픽 터미널도 개발, 그래픽터미널 시장에 본격 진출할 예정이다.

–미, HTS
국내 상륙 시도

미국 신문제작시스템 전문업체인 하이테크널로지솔루션(HTS)사가 국내 시장진출을 시도하고 있다.

관련업계에 따르면 "여호와의 증인"이란 종교단체의 계열사인 하이테크널로지솔루션사는 미국 체이스맨하턴은행을 통해 금성, 삼성, 쌍용, 대우, 현대, 선경 등 6개 대기업에 자본투자 및 시스템 판매에 대한 합작회사 설립을 검토하는 등 국내 진출을 서두르고 있다.

HTS사는 이를 위해 마케팅매니저를 파견, 시장조사를 포함한 합작회사 설립 검토를 이미 끝마친 것으로 알려졌다.

HTS사가 국내 공급할 신문제작시스템은 MPS1000으로 인텔80286칩을 사용한 16비트 퍼스컴이면 기종에 상관없이 사용가능하며 한글은 물론 희랍어, 소련어 등 103개국어가 지원가능하여 신문사를 비롯 잡지사, 일반 기업체 사내출판에 적합하다.

한편 합작회사 제의를 받은 6개 기업가운데 금성소프트웨어를 통해 CTS 시스템개발에 참여하고 있는 금성그룹이 관련계열사 책임자회의를 갖는 등 시스템 공동개발 및 합작투자에 긍정적인 반응을 보이고 있는 것으로 전해졌다.

–삼보컴퓨터
마켓팅 활성화 위해 T.S.M 신설

삼보컴퓨터는 소프트웨어 마케팅부분의 활성화를 위해 트라이젬 소프트웨어 멤버쉽 제도를 운영한다.

이는 그동안 운영해오던 트라이젬 소프트웨어 그룹의 T.S.G협력업체와 트라이젬 전문대리점 등의 응용 소프트웨어의 개발력을 향상하고 업종별, 전문분야별로 세부적인 기술지원을 함으로써 장기적으로는 영업에 실질적인 지원이 이루어질 수 있도록 하기 위한 것이다.

–삼보컴퓨터
제닉스 스크린에디터 개발

삼보컴퓨터(대표 : 김종길)는 최근 국내에서는 최초로 이미 DOS에서 사용중에 있는 2바이트 한글 / 한자 스크린에디터인 TGEDIT를 제닉스상에서도 사용할 수 있는 "제닉스용 TGEDIT"를 개발하는데 성공하였다.

그동안 제닉스에서는 에디터로 VI, ED를 사용해 왔으나 2바이트 한글 / 한자는 사용할 수 없었다. 이러한 이유에서 N바이트 한글을 처리하여 사용함으로써 DOS와 데이터 호환이 불가능하였다.

그러나 TGEDIT 제닉스는 DOS에서 작성된 프로그램이나 데이터를 제닉스에서도 사용할 수 있게 하여 데이터의 호환이 가능하게 되었다.

이에 따라 제닉스용 TGEDIT를 이용하면 제닉스에서 사용되는 터미널에서 2바이트(조합형, 완성형) 한글 / 한자(4888자) 및 특수문자의 사용이 가능하게 되고 DOS와의 데이터호환과 함께 제닉스 상에서 자유롭게 TGEDIT 화일작성, 추가, 수정, 삭제 등의 편집이 가능하게 된다.

뿐만 아니라 이번에 개발된 제닉스용 TGEDIT 버전 1.00은 DOS에서의 TGEDIT의 키값이나 기능을 똑같이 함으로써 기존의 TGEDIT 사용자들도 별 어려움없이 사용할 수 있으며, 프린트도 호스트나 터미널에서 선택하여 사용할 수 있도록 하였다.

–삼성전관
NEC 최신기종 도입, 공급에 나서

삼성전관이 금융기관, 유통업계, 중소기업의 전산화용 호스트컴퓨터로 적합한 중형컴퓨터를 발표하는 등 범용컴퓨터사업을 강화하고 있다.

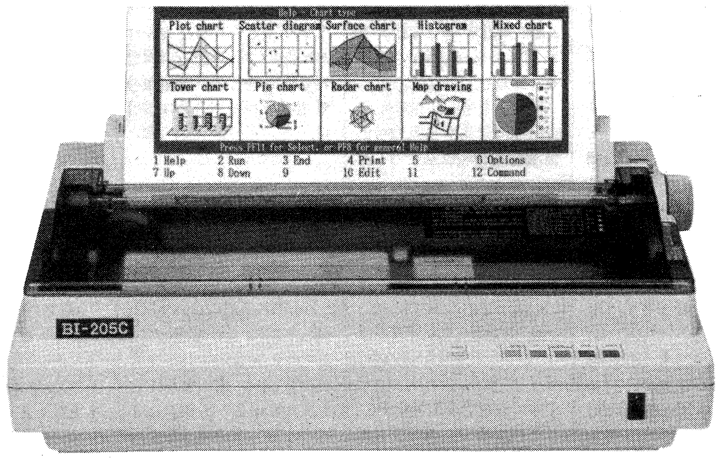
삼성전관(대표 : 김정배)은 일본NEC사가 최근 3만게이트 CMOS커스텀 VLSI 로 만들어진 마이크로프로세서를 채용해 개발한 중형컴퓨터 ACOS 3300 시리즈를 도입 공급에 나섰다.

주기억용량을 최대 60MB까지 확장할 수 있는 이 컴퓨터는 자체 고장진단전용 마이크로프로세서를 내장 사용자가 이상부분을 쉽게 찾아낼 수 있으며, 자동운전기능을 지니고 있어 컴퓨터를 어려움없이 운용할 수 있는 것이 특징이다.

이 컴퓨터는 특히 데이터의 통신능력이 최대 512K bit / sec로 고속처리가 가능한 것은 물론 630대의 단말기를 연결 동시에 사용이 가능해 LAN 구축에 적합하다.

–대광컴퓨터
IBM 5550용 컬러프린터 개발

대광컴퓨터(대표 : 박종근)는 최근 IBM5550 / 5540 컴퓨터에 연결해 사용할 수 있는 그래픽 한글 · 한자용 컬



러프린터 BI-205G를 개발하여 시판에 들어 갔다.

24핀 도트매트릭스 임팩트방식을 채택한 이 컬러프린터는 4색의 리본으로 7가지 색을 프린팅할 수 있도록 설계되어 있으며 인쇄속도가 한글·한자의 경우 초당 135자 영문자의 경우는 227자를 인쇄할 수 있는 고속인 것이 특징이다.

–삼성데이타시스템
VSP패키지 공급

삼성데이타시스템(대표 : 임경춘)은 IBM시스템 판매사업을 강화하기 위해 최근 미 소프트웨사와 IBM전용 데이터 액세스관리시스템인 VSP에 대한 독점공급계약을 체결, 본격 시판에 나섰다.

이 패키지는 IBM 데이터베이스에 내장되어 있는 각종 정보를 VSAM방식으로 구성할 수 있다. VPS패키지는 데이터영역을 탄력적으로 한정할 수 있을 뿐아니라 데이터검색 및 백업, 데이터액세스 기능을 상호 유기적으로 설치할 수 있다. 또한 오류데이터를 손쉽게 수정, 보관할 수 있어 VSAM데이터의 효율적인 운용은 물론 시스템 운용비용 및 시간을 절약할 수 있다.

–삼성전자
90평 규모의 상설전시장 개장

삼성전자(대표 : 강진구)는 일반 소비자들이 첨단 정보통신제품들을 직접 접하고 스스로 조작을 통해 보다 쉽게 이해할 수 있도록 꾸민 "정보통신 상설전시장"을 강남구 역삼동 소재 장우프라자빌딩에 개장하였다.

90평 규모의 이 전시장에는 전화기, 팩시밀리 등의 단말기기와 컴퓨터, 화상통신시스템, 가정자동화 시스템 등 각종 정보통신 제품들이 일반소비자가 직접 작동할 수 있도록 전시되어 있다.

전시장 관람은 평일 오전 9시30분부터 오후 6시까지, 토요일은 9시30분부터 오후 3시까지 이다.

–삼성전관
컴퓨터 관련기기 사업 강화

삼성전관(대표 : 김정배)은 컴퓨터 관련기기 사업을 강화하고 있다.

일본의 NEC사 제품을 국내에 공급하고 있는 삼성전관은 컴퓨터사업 확대일환으로 그동안 수출해오던 모니터의 시판에 이어 컴퓨터 관련제품을 다양화, NEC의 산업용터미널, POS(판매시점 관리시스템)등도 도입하여 판매할 것으로 알려졌다.

또한 소프트웨어 지원부서를 강화하고, 다양한 응용프로그램의 한글화에 주력할 방침이다. 이같은 컴퓨터관련 사업확대는 그동안 삼성전관에 제품을 공급해오던 삼성데이타시스템이 IBM과의 합작투자로 IBM제품만을 공급하게 됨에 따라 이분야의 강화가 불가피했기 때문으로 보인다.

한편 삼성전관은 컴퓨터 관련사업 강화책의 일환으로 최근 중형컴퓨터인 ACOS3300 시리즈를 내놓았다. 이 시리즈는 크기와 가격이 미니급이면서 대형컴퓨터의 성능을 발휘하는 중형컴퓨터로 통합 사무자동화(OA) 시스템 구축에 적합한 기종이다.

–삼성HP
네트워크 컨설팅부서 신설

삼성휴렛팩커드(대표 : 리차드 딕 워밍턴)가 네트워크 컨설팅사업에 본격 참여하였다.

삼성휴렛트커드는 컴퓨터 영업 강화의 일환으로 네트워크 컨설팅사업을 벌이기로 하고 최근 LAN-WAN 등 네트워크에 대한 컨설팅 및 지원업무를 전담할 네트워크 컨설팅부서를 별도로 신설했다.

삼성휴렛팩커드는 이 컨설팅부서를 통해 부서간 또는 회사간 컴퓨터 네트워크을 구축하고자 하는 유저들에게 네트워크 구축에 따른 정보와 네트워크 유지관리 등을 지원해 자원의 공유 및 정보교환을 효율적으로 이용하여 생산성을 향상시키도록 할 계획이다.

삼성휴렛팩커드가 네트워크 컨설팅 사업에 참여하게 된 것은 최근 컴퓨터의 급속한 보급과 함께 네트워크 활용 요구가 증대되고 있으나 컴퓨터 사용자들이 네트워크에 대한 인식 및 전문지식이 부족한데다 컴퓨터 공급업체들의 지원미비로 실행에 어려움을 겪고 있어 이를 효율적으로 이용할 수 있게 함과 동시에 컴퓨터영업을 강화하기 위한 것으로 풀이된다.

-삼성전자
경기호조에 힘입어 메모리 생산량 확대

국내 반도체생산업체들이 올해들어 메모리생산을 지속적으로 늘리고 있는 가운데 특히 삼성전자는 1M D램에 주력하여 지난해 3/4분기보다 2배 가까이 물량을 늘리고 있다.

관련업계에 따르면 삼성전자, 현대전자 등 반도체 생산업체들은 올해 256K D램, 1M D램, 256KS램 등 메모리제품 생산량을 지난해보다 꾸준히 늘리고 있다.

국내업체들이 이처럼 메모리 제품 생산량을 꾸준히 늘리고 있는 것은 반도체 경기가 아직은 호조를 보이고 있고 수율향상 및 메가라인 증설 등으로 인해 생산능력이 점차 늘고 있기 때문으로 풀이된다.

삼성전자는 1M D램을 지난해 9월 130만개를 생산하고, 지난해 연말까지 매월 10만개씩 늘려왔으나 올해들어 지난해 연말에 완공한 제3라인이 본격 가동에 들어감에 따라 올해 1월 들어 250만개로 대폭 늘리고 있다.

-STM
전산실 수탁관리사업 확대키로

STM(대표 : 김영태)은 컴퓨터를 비롯, 소프트웨어 전문인력등 제반 정보처리에 필요한 자원을 확보해 놓고, 장기계약에 의해 전산업무를 대행해 주는 전산실 수탁관리사업이 고부가가치 사업으로 각광받게 됨에 따라 이달초 금성히다찌의 M680D대형컴퓨터를 설치한데 이어 내달초 IBM 대형 컴퓨터인 IBM3090-400E를 추가 도입키로 하고 구체적인 실무작업을 벌이고 있다.

이번에 설치될 컴퓨터는 데이터처리 속도가 57.3MIPS이며 CPU메모리용량이 384MB인 초대형 시스템으로 프로그램개발은 물론 전산처리서비스에 활용된다.

존슨리서치
학습용 소프트웨어 '코스웨어' 개발

초·중·고등학교의 전산교육실시에 힘입어 그동안 일부대학교나 연구기관 등에서 제한적으로 실시되던 컴퓨터 이용 교육이 크게 확산될 조짐을 보이고 있는 가운데 국제전산, 연합시스템, 존슨리서치, SKC 등 컴퓨터메이커들이 컴퓨터를 이용, 학습효과가 뛰어난 교과과목을 중심으로 한 코스웨어 소프트웨어개발에 박차를 가하고 있다.

특히 그래픽이나 데이터통신 등 컴퓨터 하드웨어 관련기술과 프로그래밍 언어 등 소프트웨어 기술이 점차 고도화되고 있고, 컴퓨터 교육수준이 퍼스컴대중화 시대에 발맞춰 전반적으로 향상되고 있어 컴퓨터를 이용한 학습용 소프트웨어개발이 더욱 추진될 전망이다.

존슨리서치(대표 : 이병철)는 자사제품인 2350한글+에 데이터베이스기능을 첨가하여 영어나 한글 단어를 손쉽게 검색할 수 있는 "학생용 영한, 한영사전" 프로그램을 개발하여 시판에 나섰다.

16비트 IBM XT/AT 호환기 및 PS/2에 운용가능한 이 코스웨어는 학생이 자유자재로 단어를 추가·변경·삭제할 수 있는게 특징이다.

-쌍용컴퓨터
대폭적인 인사이동 단행

쌍용컴퓨터(대표 : 최병환)는 임원급 인사 이동을 단행하였다.

이번 인사이동으로 김용서, 이성우 상무이사가 전무로 승진되었고, 강용중, 황규선 이사가 상무이사로 각각 승진되었다. 교육센터 부장인 효동균씨는 이사로 승진하였고, 노중호 전무이사는 '대표 컨설턴드'로 전보발령이 났다.

이번 인사이동의 특징은 승진 일변도로 단행 되었는데 지난 2~3년간의 사업확장과 매출액신장 기여도에 따른 것으로 풀이된다.

-한국데이타통신
종합정보통신센터 대덕에 설립 추진

한국데이타통신(사장 : 이응효)은 정보통신기반을 지방에도 조성하고 이용의 활성화를 위하여 충남 대덕에 첨단 컴퓨터통신시설을 갖춘 텔리포트형 종합정보 통신단지를 조성하기로 하고 최근 1만8천여평의 부지를 확보했다.

한국데이타통신은 이 부지에 각종 정보서비스를 제공할 수 있는 시설인 데이터의 교환·제공을 위한 광섬유케이블망, 이들 설비의 운영을 콘트롤하는 텔리센터(Telecenter), 위성통신용 지상안테나 시설을 갖출 계획이다.

-한국 비지네스서비스
주변기기 관리 소프트웨어 독점 공급

한국 비지네스서비스(대표 : 김정근)는 거래선인 미국듀퀜시스템즈가 (미)모리노사에 합병됨에 따라 모리노사의 종합정보시스템 관리용 소프트웨어인 "믹스"패키지를 국내독점 공급하기로 했다고 밝혔다.

이 소프트웨어는 MVS 및 VSE를 오퍼레이팅시스템으로 채택한 대형컴퓨터에 부착되어 있는 콘트롤러, 모니터 등 주변기기를 종합, 관리해 주는 시스템으로 대형컴퓨터의 데이터 처리 속도를 자유자재로 조절하여 부하를 줄여주며 각종 주변기기의 운용시간을 조정, 전산비용을 절감할 수 있다.

-쌍용컴퓨터
"비픽스"관련 기술도입

쌍용 컴퓨터(대표 : 최병환)는 최근

과학기술처로부터 종합 생산관리시스템인 비픽스관련 기술도입을 승인 받았다.

쌍용컴퓨터는 이에 따라 기술제공사인 미국SAA사로부터 "비픽스"패키지에 대한 소스프로그램 등 각종 기술자료를 제공받아 국내 실정에 맞도록 재설계할 수 있게 되었다.

제조 및 유통업체들의 합리적인 경영관리를 위해 개발된 이 소프트웨어는 생산, 판매, 재무, 경리 등 20여개의 모듈이 상호 유기적으로 구성되어 있으며, 기업의 규모, 전산화 추진계획에 따라 모듈별로 구분 단계적으로 도입할 수 있는 것이 특징이다.

-한국신용평가(주)
온라인 DB사업 강화

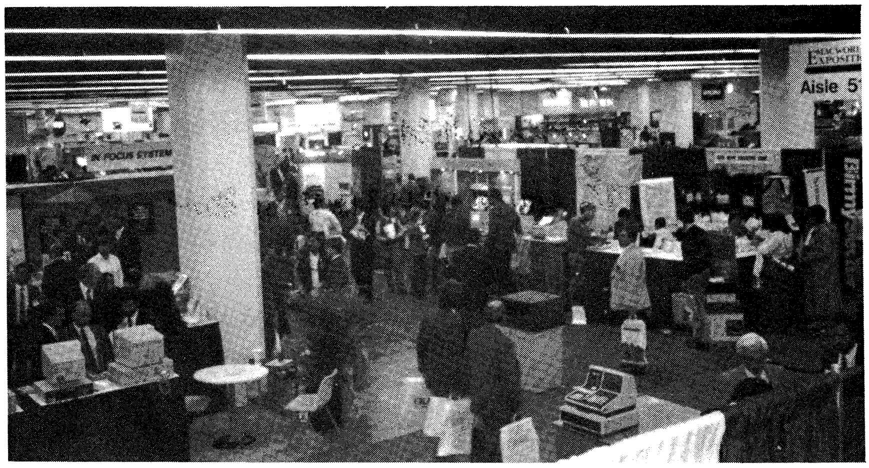
한국신용평가(주)는 지난해부터 상용서비스에 들어간 온라인 신용평가정보서비스(BRIFS)의 확산을 위해 DB내용 및 시스템을 보강하는 등 온라인 DB 사업을 강화할 것이라고 밝혔다.

동사는 한국증권전산(주)과 계약하고, 구축중인 상장사 증권정보를 비롯한 투자정보 구축작업을 완료하고 3월부터 서비스할 계획이며, 추후 이들 투자정보를 그래픽으로 파악할 수 있도록 보강해 나갈 것이라고 한다.

이와 함께 IBM 3725 통신제어장치를 도입하여 64포트를 수용할 수 있도록 하고 이용자 단말기로 범용 PC를 이용할 수 있도록 시스템을 보강할 방침이다.

-(주)엘렉스컴퓨터
대리점 사장단 Mac World 참관

엘렉스컴퓨터(대표 : 이윤기)의 대리점 사장단과 엘렉스컴퓨터 직원 등을 포함한 27명은 지난 1월 20일부터 22일까지 미국 캘리포니아주 샌프란시스코에서 열린 Mac World 89에 참관하였다.



총 467개사의 미국업체가 참가한 이번 Mac World 89는 전체적으로 데스크탑 퍼블리싱, 엔지니어링, 커뮤니케이션, OA, 프리젠테이션 등 5가지로 분류되어 진행되었다.

엘렉스컴퓨터는 컴퓨터가 첨단 제품만이 아니라 누구나 쉽게 접근할 수 있는 시스템이 되도록 하기 위해 한국시장의 성격에 맞는 시뮬레이션 프로그램개발에 주력할 것이라 한다.

-중소 소프트웨어업체
호텔 관리용 S/W 개발

호텔업무의 전산화열기가 한층 고조되고 있다. 객실 200개 안팎의 중소형 호텔관리용 소프트웨어가 개발되어 그동안 외국패키지의 수입판매에 의존하던 호텔 관리시스템시장이 새로운 국면을 맞고 있다.

호텔관리시스템은 타업종에 비해 업무가 복잡해 시스템설계 및 개발이 까다로운 편이다.

여기에 신천일렉트로닉, 금성컴퓨터하우스, 바이트전자 등 중소 소프트웨어 업체들이 퍼스컴용 호텔 관리시스템을 개발하여 영업활동을 벌이고 있다.

특히 이들 국산제품은 각 호텔 고유의 성격이나 호텔영업 및 일반관리업무에 따라 시스템을 손쉽게 수정, 보완할 수 있으며, 가격이 외국제품보다 저렴해 가격경쟁력이 뛰어난 것으로 알려졌다.

또한 16비트 퍼스컴에 사용 가능하도록 설계되어 있어 업무확장과 함께 퍼스컴 LAN을 구축할 수 있는 장점이 있다.

-컴퓨터월드
네트워크사업 참여

소프트웨어 전문업체인 컴퓨터월드(대표 : 박상훈)는 네트워크사업에 나서기로 하고 미국 LAN전문업체인 3콤사의 국내 독점공급선인 IMD코리아와 LAN시스템에 관한 독점 딜러계약을 체결 시스템공급에 나섰다.

이에 따라 그동안 3콤 네트워크시스템을 독점 공급해 오던 ICS는 이 사업에서 완전히 손떼게 됐다.

컴퓨터월드가 공급할 3콤 네트워크 시스템은 기존의 LAN시스템보다 데이터처리 속도가 빠르고 MS-DOS상에서 운용되는 각종 응용소프트웨어를 변환없이 그대로 사용할 수 있는 것이 특징이다. 또한 퍼스컴간의 네트워크는 물론 미니컴퓨터나 메인프레임과도 쉽게 연결할 수 있다.

컴퓨터월드는 3콤 본사와 공동으로

정기 기술세미나를 개최하는 한편 사용자 상호간의 정보교환을 위해 유저 그룹을 설치 운영할 방침이다.

-(주)통일
한국PTI 인수해 HDD업계 진출

국내 유일의 3.5인치 HDD 생산업체인 한국PTI가 통일교 재단인 (주)통일로 넘어간다.

관련업계에 따르면 한국PTI는 (주)통일에 전체 자본금의 50%이상을 증자시켜줄 것을 조건으로 매각, 실질적인 대주주가 바뀌는 방식을 채택함으로써 (주)통일에 의해 인수될 것으로 보이며, 현재의 주주인 일부 국내 단자회사들과 일본측 파이브 시스템사 등은 이 회사의 경영에서 손을 뗄 것으로 알려졌다.

이에 따라 현재 자본금이 20억, 부채가 30억원으로 되어 있는 한국PTI는 (주)통일이 자본을 투자할 경우 총자본금이 80억원으로 늘어나게 된다.

한편 (주)통일측은 한국PTI를 계열화시켜 운영할 것인지 또는 이 회사의 전자부에 조직을 흡수 통합할 것인지는 아직 결정한 바 없다고 밝혔다. 그러나 한국PTI가 생산해 온 주요제품인 3.5인치 20, 30MB 및 앞으로 시장수요가 급격하게 신장될 40MB 모델을 계속 생산하게 될 것이라고 전했다.

종교재단인 통일교가 100% 출자한 (주)통일은 머시닝센터 등 공작기계를

● 화제의 현장 ●

컴퓨터를 이용하여 헤어스타일을 창조한다.

최근 컴퓨터의 보급이 급격히 확산되면서 모든 분야에서 그 이용이 활발히 진행되고 있는 가운데, 특히 컴퓨터와 영상기술이 혼합된 이미지 시스템기술이 발전함에 따라 컴퓨터 그래픽과 영상을 이용한 신기술이 개발되어 보급되고 있다.

이러한 형태의 이용은 국내에서는 메이크업에 관련된 소프트웨어가 개발되면서 몇몇 화장품 회사에서 고객 서비스용으로 설치되어 사람들의 관심을 끌어 왔다. 그러나 이러한 단계는 메이크업이나 의상에 국한된 분야이었고 전체적인 패션을 창조해 내는 데는 효과를 미치지 못했다.

이런 국내의 현실 속으로 새롭고 다양한 기능을 갖춘 시스템이 도입되어 국내에서도 최초로 청담동의 박준 미장(전화 : 511-1414)에서 실용화하여 미용업계의 새로운 관심을 불러 일으키고 있다.

이 시스템은 입력장치로는 그래픽 테블릿, 비디오 카메라와 키보드이며, 16비트의 본체와 출력장치는 RGB 모니터, 프린터, 포라로이드 플레이트가 그 기본시스템으로 구성되어 있다.

이 시스템은 컴퓨터에 입력된 자료를 이용하여 자신의 모습에 변화를 주지않고도 자신이 변화된 모습을 사진으로 받아 볼 수 있도록 한 일종의 시뮬레이션 시스템이다. 우선 대상 인물에 대하여 영상을 비디오 카메라를 이용하여 포착한 다음 컴퓨터에 입력하면 입력된 영상은 재구성을 위한 상태로 전환되고 메뉴선택의 운영기법이 사용되어 라이트 펜과 키보드를 통하여 고객은 자신에게 맞는 스타일을 선택할 수 있다. 이때 선택한 스타일에 약간의 변화를 주고 싶다면 스타일을 새롭게 디자인할 수 있다. 이렇게 하여 완성된 새로운 스타일의 모습을 포라로이드 플레이트를 통하여 직접 받아 볼 수 있게 된다. 또한 새로운 스타일을 고객, 잡지, TV, 비디오 등에서 영상으로 포착하여 새로운 스타일로 개발할 수도 있다.

이러한 형태로 작업이 수행되고 있는 이 시스템의 특징은 헤어 스타일, 메이크업, 칼라 분석, 그리고 안경, 귀걸이, 목걸이 등 각종 악세서리를 기존의 입력상태에 첨가할 수 있어 전체적인 패션의 조화를 이루도록 돕고 있는 응용디자인과 고객관리 등 다양한 기능을 지니고 있다는 것이다.

여기에 광처리 부분을 개선시켜 영상의 굴절과 크기의 자유로운 조절이 가능하여 좀더 세세한 변화를 줄 수 있게 한다면 이 시스템은 더 강력한 시스템이 될 수 있을 것이다.

그러나 여기서 시각을 달리 보아 대중적인 면을 고려해 볼 때 이 시스템은 우리가 쉽게 드나들 수 있는 평범한 미용실에는 대중적으로 보급되기 어렵다는 아쉬움을



전문으로 제작 판매해 온 중견기업으로서 지난해부터 정보산업체 진출의사를 강력하게 비쳐 왔었다.

-테레비디오컴퓨터

컴퓨터 생산에 표면실장기술 이용

SMT(Surface Mount Technology) 기법을 이용한 컴퓨터의 생산이 국내에서 처음으로 테레비디오컴퓨터에 의해 실현되어 업계의 관심을 끌고 있다.

컴퓨터 전문업체인 테레비디오컴퓨터(대표:규빈 필립황)는 최근 SMT 기법을 실현할 수 있는 PCB조립자동화 생산설비를 갖추고 이를 이용해 16비트 및 32비트 퍼스컴과 CRT터미널을 생산하고 있다.

표면실장기술이라 불리는 SMT기법은 PCB에 각종 부품을 장착할 때 PCB 표면에 부착하여 기존 부품이 차지하는 공간을 최소화할 뿐 아니라 제품의 신뢰도를 높여주는 새로운 PCB조립기술로 미국·일본·유럽 등 선진국에서는 이미 이 기법을 이용해 각종 전자제품을 생산해오고 있다.

SMT기법을 이용할 경우 제품의 생산성향상과 품질고급화를 실현할 수 있게 된다.

테레비디오가 컴퓨터제품 생산에 SMT 기법을 도입하게 된 것은 최근 세계 퍼스컴기술의 소형, 고급화 추세에 대응하고 원화절상, 임금인상 등으로 인해 약화되고 있는 수출 경쟁력을 높이기 위한 것으로 보인다.

-한국데이타통신

다중화장치 사용기준 마련

데이터통신은 특정 통신회선을 훨씬 경제적이고 효과적으로 이용할 수 있도록 다중화장치 사용기준을 마련하기로 했다. 그리고 특정통신 이용약관을 개정하고, 다중화장비를 구입하지 않은 이용자를 상대로 회선분기 서비스를 제공하기로 했다.

데이타통신이 마련한 다중화장치 사용기준을 보면, 접속사용회선의 속도는 9600bps이하로 하고, 전기통신설비의 기술기준에 관한 규칙 제10조에 의한 전기통신 기자재의 시험검사를 받은 것에 한해 접속되도록 하며, 회선청약시 사용자에게 품명·수량·용량 등의 신고를 의무화하고 있다.

데이타통신은 특정 통신회선서비스가 실시됨에 따라 회선망자체에 MUX 장비를 갖추고 1차로 2월부터 9600bps 이하의 회선에 한해 분기서비스를 제공할 계획이다. 이에 따라 MUX를 갖고 있는 그룹 VAN 및 역무제공업자는 자사의 MUX를, MUX를 갖지 못한 역무제공업자나 일반사업자는 데이타통신의 설비를 이용할 수 있게 됐다.

-현대건설

소프트웨어산업 진출 추진

공장 및 사무자동화 붐에 힘입어 소프트웨어산업이 전반적으로 안정기에 접어들고 있는 가운데 현대건설 등 국내 대형 건설업체들이 소프트웨어산업에 진출을 서두르고 있다.

이같은 현상은 건설업의 채산성이 급격히 떨어지고 있는데다가 건설업계 전산화열기가 점차 고조되고 있기 때문으로 풀이된다.

현대건설은 도급관리, 공사일정, 노무, 현장관리 등 건설현장에서 생기는 각종 정보를 신속하게 처리하기 위해 이미 개발되었던 건설종합관리시스템을 중소건설업체에 맞도록 제품화하기 위해 시장조사에 나섰다.

-현대전자산업(주)

주전산기 톨러런트 국산화

행정전산망용 주전산기인 톨러런트가 국산화되어 공급되기 시작했다.

현대전자산업(주)외 금성사, 대우통신, 삼성전자 등 4개사는 한국데이타통신과 1차 수요분 29대에 대한 공급계약을 체결하였다.

'83년 8비트 PC 국산화 이후에 이루어진 이번 주전산기 국산화는 '87년 4월 국가기간전산망 조정위원회 행정망용 주전산기 국산개발 공급계획이 의결됨에 따라 1단계로 톨러런트 시스템사로부터 관련기술을 도입, 체제를 구축하였으며, 2단계로 독자적 수퍼미니컴퓨터 개발을 할 예정이다.

내포하고 있다. 그것은 시스템의 소프트웨어의 개발이 국내에서는 이루어지지 않고 외국에서 이루어졌기 때문에 값비싼 로열티를 지불해야 한다는 난점이 있다. 그러므로 이 시스템을 구비하려면 어느 정도 경제적인 조건(약2천5백만원)을 갖추어야 한다. 이러한 문제점은 곧 고객이 그 비용의 일부를 약간이나마 부담해야 하는 껄끄러운 결론(고객서비스의 차원이라는 측면에서 벗어날 수 있다.)에 도달할 수도 있기 때문이다.

이러한 점에서 볼 때 정보화 시대에 걸맞게 생활주변의 모든 것이 컴퓨터를 이용하여 보다 편리한 도구가 되도록 개발된 외국의 첨단 기기를 도입하여 새로운 기술을 이용하고 있다는 것도 의미있는 일이지만 값비싼 로열티를 지불하는 대신에 그러한 개발의 발원들이 국내에서 출발할 수 있도록 인적, 물적 요건을 조성하는데 투자의 기반을 마련하는 것이 가장 시급한 문제이다.

그렇게 된다면 누구나 쉽게 그리고 부담없이 자신의 모습에 변화를 주고 싶을 때 컴퓨터를 통하여 구체적인 창조를 해내어 누구나 자신의 개성을 살려낼 수 있다. 이제 이러한 시도가 우리의 가까운 주변에서 일어날 수 있으리라 전망해 본다.

국외소식

미, 드라이브시장
디스크 어레이 등장으로 파문일어

최근들어 하드디스크 드라이브(HDD) 시장에 새로운 역사가 창출되고 있다. 소규모의 업체들이 주동이 되어 벌이고 있는 이 작업은 기존의 드라이브 개념을 뛰어넘는 디스크 어레이라는 새로운 개념이다.

DA(Disk Array)란 여러대(보통 6대)의 HDD를 전기적으로 연결(각 드라이브의 스핀들(Spindle)을 동기연동)하며 한 콘트롤러의 코맨드하에서 전 HDD가 데이터를 동시에 로깅하는 드라이브기구를 말한다. DA는 단일의 HDD에서는 어려운 시스템전체의 데이터 파괴방지(무정지)를 실현하는 것 외에 I/O데이터의 전송속도를 대폭으로 향상시킬 수 있다.

이 DA시장을 지금 몇개의(주로 실리콘밸리) 신흥기업군이 주동이 되어 선도하기 시작하였고(표 1), 대기업인 메인프레임/미니컴 메이커가(IBM, DEC, Tandom, CDC 등) 스스로의 참여시기를 호시탐탐 노리고 있다. 이와 같이 소기업이 주동이 된 요인에는 두가지의 이유가 있다.

그 첫번째는 종래 메인프레임이나 미니컴의 데이터처리 영역이였던 대용량의 데이터 블록전송을 요하는 어플리케이션(통신, 데이터베이스, 그래픽 화상처리, CAD/CAM 그리고 DTP 등)이 32비트 PC의 사정거리에 들어온 것이다.

두번째는 HDD가 용량, 전송속도, 신뢰성, 코스트 등의 면에서 대형(8인치 이상)에서 5.25인치, 3.5인치로 급속히 소량화가 진전하고 있는 것을 들 수 있다. (1988년 HDD출하수량 1470만대 데이타퀘스트사 조사중에서 이미 54%가 3.5인치, 46%가 5.25인치의 시장구성으로 되어 있읍니다(표 2, 표 3)).

그렇더라도 HDD단일체에서는 고속, 대용량 어플리케이션을 구사하려면 스피드부족으로 데이터의 신뢰성(홀드 톨러런트/무정지)을 보증할 수 없다. CPU가 년 40%씩 고속화하고 있음에도 불구하고 외부기억장치(HDD)는 CPU에 비해서 수백배 정도 늦다(그 결과 CPU가 10배 고속화되어도 컴퓨터 시스템전체는 5배 밖에 빨라지지 않는다)는 이야기이다. 즉 I/O의 병목현상이 발생하고 있는 것이다.

이 HDD의 문제점을 해결하기 위해서 등장한 DA제품의 특징(이제 공급이 개시되었을 뿐)을 선행메이커 Pacstor 사의 DA상위기종(Integra Ⅲ)에서 보면 다음과 같다.

이 DA기종은 100M바이트 HDD를 12대 연결하여 80386 CPU를 사용하는 PC를 16대까지 제어한다. 그 개요는 다음과 같다.

- 액세스시간-4㎳초
- 데이터 전송속도-1.25M바이트/초
- MTBL(평균데이터 로스시간)-75만 시간
- Novell, 3Com 각 네트워크, PS/2, PC/AT, Unix의 각 OS와 호환성 있음
- 주요 어플리케이션-트랜잭션처리, DTP, CAD/CAM, 통신, 군용프로그램, 프로세스 콘트롤 등
- 생산담당회사-Key Tronic사(키보드 전문 대기업)

DA시장에 대해서 대형 컴퓨터메이커간에는 신중론도 있다. 그 대표적인 소리는『메인프레임에서 PC로, 그리고 통신 및 그래픽스관련의 어플리케이션이 갖추어지려면 앞으로 2년이 걸린다.

〔표 1〕 DA시장의 주요 메이커 7사

메 이 커	소 재 지	현 황
Pacstor Inc.	Los Gatos, CA	• DA 서브시스템(Integra)을 공급중
Micropolis Corp.	Chatsworth, CA	• 공급(테스트 판매)중—380M바이트 데이터용에 5.25" HDD 4대와 패리티용 HDD 1대를 연결(SCSI 사용)
Auspex Inc.	Santa Clara, CA	• DA 시장진출계획중
Priam Corp.	San Jose, CA	• 최근 DA생산개시 예정
Intellistor(후지쯔(아메리카자회사)	Longmont, CO	• DA 시장진출 검토중
Amperif Corp.	Chatsworth, CA	• DA 시장진출 검토중
Sequent Computer Systems Inc.	Beaverton, OR	• DA 시장진출 검토중

〔표 2〕 HDD의 세계 시장규모 추이

(자료 : Disk/Trend Inc.) (출하금액기준 1억달러)

1986	1987	1988	1989	1990
147	181	209	241	267

〔표 3〕 87년 메이커별 HDD 시장점유율(%)

메이커	5.25인치	3.5인치
Seagate	54.9	1%이하
Miniscribe	11.4	21.6
CDC	7.3	—
Micropolis	6.4	—
Quantum	2.4	—
Maxtor	2.1	—
IBM	—	20.6
Tandon	—	18.8
JVC(일본 빅터)	—	8.2
Conner Peripherals	—	7.4
기타	15.5	22.4
출하수량	640만대(100%)	360만대(100%)

(자료 : Dataquest)

그 사이에 타이밍을 가늠하여 진출하겠다』(Seagate Technology) 『현재 모든 어플리케이션에 대처할 수 있는 DA는 아직 공급되고 있지 않다』(CDC) 등의 의견이다.

DA의 불안정(무용론)에는 나름대로의 두가지 근거가 있다. 첫째는 현재 상황에서 대부분의 PC어플리케이션에서 무정지기능은 불필요하며, 때문에 DA는 불필요하다는 생각이다. (물론 OS/2하에서 쓰이는 386데이터베이스 서버가 등장하면 상황은 바뀐다.)

두번째로 DA는 특수한 에러 콜렉션 소프트웨어, 마스터 콘트롤러, 스핀들/액추에이터 동기기구가 필요하게 되기 때문에 그만큼 메가당단가가 늘어난다는 의견이다(3.5인치/5.25인치 HDD에서는 3~5달러/OEM), 이러한 시장배경하에서 메인프레임, 미니컴 메이커군은 FDD시대와 마찬가지로 DA시장의 성숙화를 조용히 바라보고 있다. 신흥 DA메이커군이 어떤 방향으로 나갈 것인가. 그것은 2년뒤쯤에나 판가름날 것이다.

애플사

Mac II로 멀티미디어 시장진출

얼마전 애플사가 주최한 『Interactive Multimedia Develoment Comference』는 참가자가 약 700명(반은 동사사원이였다고 한다)으로 비교적 소규모였지만 동사가 멀티미디어분야에 참여하는 것을 정식으로 발표함과 동시에 Mac II를 멀티미디어 시스템의 출발점으로 위치짓는 점에서 매우 흥미깊은 것이었다.

이제까지 스테이지나 스튜디오의 음악분야에서 Mac은 폭넓게 사용되어 왔지만 멀티미디어 관련, 특히 퍼스널 컴퓨터를 사용한 비디오 이미지처리 분야에서는 Amiga쪽이 앞서 있었다.

실제로 텔레비전의 일기예보 등을 보고 있으면 Amiga로 처리한 그래픽스가 빈번하게 사용되고 있다. 또한 보다 전문분야에서는 Silicon Graphics사의 『IRIS』 등 UNIX그래픽스 머신이 주류를 이루고 있다.

그러나 Mac II가 발표되고 나서 이 머신의 멀티미디어 분야에의 진출은 눈부신 것이었다. 이 분야의 수요에 부응하도록 동사는 89년 초반 32비트 대응의 Quick Draw의 릴리즈를 예정하고 있다. 동사의 이러한 움직임에 발맞추어 제3그룹도 활발한 움직임을 나타내고 있다. Mass Micro Systems사는 Mac II용의 비디오 그래픽스 보드를 개발하여 이 컴퍼런스에서 소개하였다.

『Color Space FX』라고 불리우는 이 보드는 Mac II의 NuBus슬롯에 장착함으로서 풀 컬러, 풀 모션 비디오를 디스플레이의 윈도우내에 리얼타임으로 표시할 수 있다. 따라서 거울면효과, 줌인(zoom in)/아웃, 축소, 밀확대 등 가지각색의 처리를 윈도우내에서 비디오 이미지의 표시중에 다이나믹하게 하는 것이 가능해졌다.

이러한 처리나 보드의 드라이버조작은 하이퍼 카드의 버튼을 클릭하는 것으로 간단히 콘트롤할 수 있다.

비디오신호 입력은 NTSC(미국, 일본에서의 비디오신호 표준규격), PAL(유럽의 표준규격), SECAM(프랑스의 표준규격)을 자동으로 전환할 수 있는 포트 3개를 표준으로 장비하고 있다. 프레임의 도입과 표시속도는 비인터레이스 멀티싱크 모니터에서 최대 60프레임/초로서 보통의 비디오신호 출력에는 충분히 대응할 수 있는 속도로 되어 있다.

동사는 Mac II + Color Space FX의 시스템에서 주로 개발자나 프로의 영상기술자를 중심으로 마케팅을 전개하여 갈 계획이다.

멀티미디어지향 시스템에서 특히 주목되고 있는 것이 Hypertext이다. 오디오, 비쥬얼 그리고 인터랙티브(Interactive) 처리를 구사한 시스템의 구축에는 Hypertext의 개념이 불가결로 되고 있다.

애플사의 HyperCard는 가장 단적인 예이며, 퍼브릭 도메인을 포함한 어플리케이션은 이미 수천개가 유통되고

있다. 동사는 HyperCard를 베이스로 장래는 『Knowledge Navigator』라고 불리우는 새로운 환경을 구축할 계획이다. 그러나 현재의 HyperCard에 대해서는 아직도 불만이 많다.

불만의 대부분은 HyperCard의 윈도우 크기가 Macplus / SE의 화면에 맞추어 고정되어 있는 것과 컬러가 취급되지 않는 점에 집중하고 있다. 또 스택은 동작시키기 위해서는 HyperCard 본체를 환경으로서 장비하여 둘 필요가 있기 때문에 완전한 스탠드얼론의 어플리케이션을 만들 수 없다는 결점을 지적하는 소리도 많다.

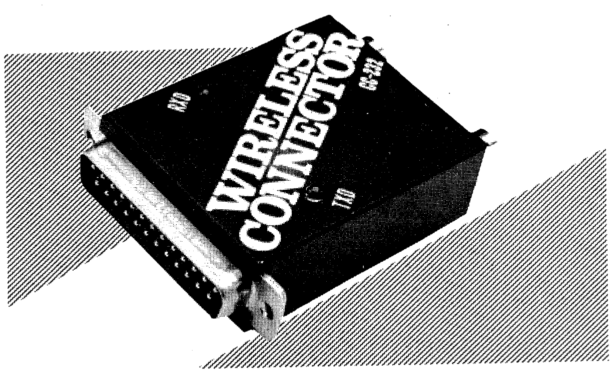
이와 같은 결점을 해결하고자 하는 노력이 애플사를 포함하여 몇개의 제 3그룹에서 이루어지고 있다. 그 하나가 Silicon Beach Software사가 개발중인 HyperCard 호환인 Hypertext 툴 『SuperCard』이다.

이 소프트웨어는 HyperTalk 스크리프트, XCMO(확장코맨드), XFCT(확장펑션) 등과 호환이다. 물론 256색 컬러 모드에도 대응하고 있고, 윈도우의 사이즈도 가변이다. 또 멀티스택처리, 즉 여러개의 스택을 동시에 화면상에 오픈하여 표시할 수 있어서 컬러 사이클링을 사용한 의사 애니메이션효과나 동사의 페인트툴 『SuperPaint2.0』 및 『Digital Darkroom』에서 작성한 그래픽스 데이터를 핸들링할 수 있는 등 오리지널 HyperCard를 대폭 능가하는 스펙을 장비하고 있다.

필요없는 기능을 없애고 스탠드얼론의 어플리케이션이 만들어지기 때문에 약250k바이트의 커널부분+어플리케이션 데이터부분에서 동작시킬 수 있다. 즉, 멀티미디어대응 어플리케이션 개발을 위한 강력한 툴로 되는 것이다.

엔 엔드 엠

선이 필요없는 무선 코넥터 발표



일본의 (주)엔 엔드 엠에서는 좁은 장소나 케이블 부설등이 곤란한 장소에 설치하기 간편한 무선 코넥터를 개발, 발표하였다. 이것은 V / UHF대의 전화를 이용하여 퍼스컴, 워드프로세서, 모뎀, 프린터간 등을 무선으로 연결할 수 있으며, 주파수는 250~400MHz대의 2파를 이용한다. 또한 채널은 5채널, 안테나는 3종이 마련되어 있으므로 네트워크의 구성도 가능하다. 통신방식은 전2중 비동기방식을 이용하며 300~9600BPS의 속도로 통신이 가능하다. 전원은 불필요하며 각 기기간을 무선으로 연결하여 상호간에 공동으로 이용할 수 있어 모뎀을 이용한 PC간의 통신시 언제든지 편리하게 이동하며 통신할 수 있는 것이 큰 장점이다. 판매가격은 2대 세트당 약 14만원정도이다.

내셔널 세미콘덕터사

영상처리용 전용칩 개발

개인용 컴퓨터사용자는 현재 레이저 프린팅이 많이 사용되고 있지만 하드카피를 위해서 아직까지 가격이 저렴한 도트매트릭스 프린터에 많이 의존하고 있다.

현재 레이저 프린터의 높은 가격은 대다수의 PC사용자에게 큰 부담이 되고 있다 . 그러나 곧 저가격의 레이저 프린터가 선보일 계획으로 되어 있어 PC사용자의 기대를 부풀게 하고 있다.

미국의 유명 반도체메이커인 NS의 시리즈 32000 마이크로프로세서 영업담당이사인 Paul Wells씨에 따르면 저가의 레이저 프린터가 2년이내에 시장에 나오게 될 것이라고 예측하고 있다.

레이저 프린팅을 비롯, 영상처리 응용제품 등에 적용되는 전용칩인 NS32CG16 에 대해 최근 아시아지역을 순방하면서 일련의 세미나를 개최하기도 한 Wells씨는 "만약 설계자가 레이저 프린터용 프로세서로 다른 제품을 선택하면 1년 혹은 1개월 정도의 설계기간이 소요될 것"이라고 말한다.

NS사의 NS32CG16 프린터 / 디스플레이 마이크로프로세서와 그것을 지원하는 칩들은 설계자들이나 제조업자들에게 이용가능하며, 영상처리 주변기기시장에 적합한 해결책을 제공할 것이라고 한다.

NS32CG16은 줄긋기, 그래픽, 패턴 확장과 복사, 그리고 데이터압축과 확장을 위한 보다 우수한 기능을 가짐으로써 현대식 영상장비에서 요구되는 모든 것을 충족시켜 준다.

이 칩의 특성은 내장된 BITBLT기능이다. 이것은 CRT스크린이나 종이에 영상을 조합하는 수백만개의 점들을 잘 처리할 수 있는 기능이다.

동사는 이 칩이 회사를 영상주변기기 구성품시장에서 선두적인 공급업체로 확장시켜 줄 것이라고 확신하고 있

Wells씨는 이러한 칩의 판매 만으로 실제 들어오는 총수익이 향후 2년내에

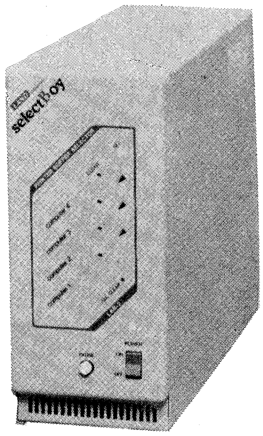
1억5천만달러를 초과하게 될 것이라며 "미래의 컴퓨터 주변기기들은 대규모의 내장된 기능을 가짐으로써 점차적으로 강력해지는 개인용 컴퓨터도 현재 일반적으로 사용되는 것보다 더 강력한 처리능력과 속도를 필요로 하게 될 것"이라고 덧붙인다.

현재 이 NS32CG16을 이용한 제품 설계에 몇몇 업체가 돌입하였으며, 곧 제품이 생산될 것이라고 한다. 동사의 이 칩은 한국·홍콩·대만·싱가폴 등을 주요 영업타겟으로 삼고 있다.

후지쯔
수퍼컴퓨터 VP2000 시리즈 발표

후지쯔는 단일CPU로 세계최고속의 4GFLOPS를 실현한 수퍼컴퓨터 『FACOM VP2000시리즈』를 발매하였다. 주요 특징은 ① 1칩 15000게이트, 지연시간 80ps의 논리LSI, 1Mbit SRAM, 61층 고밀도다층기판 등을 채용, ② 듀얼칼라 프로세서방식의 채용에 의해 보다 처리능력의 향상과 코스트퍼포먼스의 개선을 실현, ③ 메인메모리는 최대2Gbytes, 시스템메모리로서 최대 8Gbytes 를 서포트, ④ OS에 동사의 MSP와 UNIX를 채용하여 NFS나 X-Window를 서포트 등을 가지고 있다. 제품 출하는 제4/4분기 예정.

일, 랜드컴퓨터
자동 프린터 전환기 발표



랜드컴퓨터에서 버퍼가 내장된 자동 프린터전환기 『LAB-2』를 발표하였다.

동제품은 4대의 컴퓨터와 센트로닉스 규격의 프린터1대를 자동적으로 접속하는 전환기이다. 또 대용량버퍼메모리(최대:4MB)를 내장하고 있으므로 프린트 출력을 위해서 기다리던 컴퓨터의 대기시간을 대폭으로 단축해서 사무효율을 비약적으로 향상시킬 수가 있다.

최대 연결가능 컴퓨터는 4대이며 이 중 먼저 입력이 있었던 것부터 순서대로 읽어 넣을 수 있는 모드와 4개의 데이터를 균등하게 읽어 넣을 수 있는 모드가 있다.

서클동정

●경·컴·연●

전국 연합써클과 활발한 교류 계획

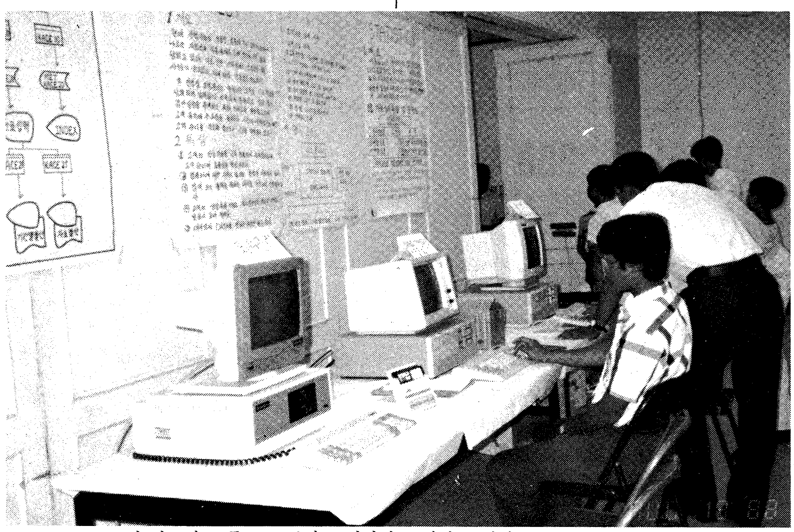
경남지구 대학의 컴퓨터 써클들이 모여 경남의 컴퓨터 문화 발전에 기여하며, 각 대학교 컴퓨터써클간의 정보교류와 기술개발을 통해 각 써클의 발전과 원활한 인간관계 유지를 목적으로 경남지역 컴퓨터 써클 연합회(경·컴·연)는 지난 87년 9월에 창립되었다.

동 연합회는 방학을 이용하여 컴퓨터 지식에 관한 내용중 필요한 부분을 택하여 학술세미나와 전시회, 체육대회 등을 실시하고 있다.

동 연합회의 회원교육은 각 학교에서 담당하고 있으며 필요한 경우 단체 견학이나 교육을 실시하고 있다. 지난 동계방학을 이용하여서는 VAN과 데이콤에서 실시한 데이타 통신, 정보화사회에 관한 강의에 참석하였다고 한다.

동 연합회는 경남대, 경상대, 진주교대, 창원대, 창원기능대, 울산대 등 6개 대학의 회원 900명으로 구성되어 있으며, 회장과 부회장, 총무, 섭외, 홍보, 기획, 편집, 연구부장 등의 운영위원들이 연합회와 관련된 모든 사항을 결정하고 각 학교에 2명의 집행위원을 두어 운영위원회가 결정한 사항을 집행하도록 하고 있다.

동 써클에 가입을 희망하는 단체는 전문대학 이상으로 학교가 승인하는 공식 써클(단 한 학교당 한 써클로 한다)이면 자격이 주어지며, 회원교에 가입되면 매학기에 2만원의 회비를 납부하면 된다.



6개 학교가 공동으로 꾸미는 전시회는 회원수 이상으로 항상 풍성하다.

홍승섭 회장(경남대 산공 3)은 회원교를 계속적으로 확대할 예정이며, 전국 각 지역의 컴퓨터 써클과 연계관계를 가져 실질적인 운영을 도모할 것이며, 경남지역의 일반 대학생들에게 정보화사회에 대한 인식을 심어주기 위한 세미나를 실시할 예정이라고 한다.

경·컴·연에 관하여 알고 싶으면 전화번호 (0551)45-5000~9 교환 2700 경남대학교 컴퓨터 써클로 하면 된다.

●서울대학교 컴퓨터 써클●

철저한 사부제도로 실력과 전통을 확립

지난 85년 3월 계산통계학과 학생들이 주축이 되어 컴퓨터 전반에 걸친 연구와 교육 및 회원의 유대강화를 통해 건전한 대학문화 발전에 기여한다는 목적으로 서울대학교 컴퓨터 써클(S.C.S.C)이 창립되었다.

동 써클의 현재 회원은 준회원 30명, 정회원 50명, 명예회원 30명으로 구성되어 있고 서울대학교 학생으로 컴퓨터에 관심있는 사람 누구나 S.C.S.C에 가입할 수 있으며, 회원은 매학기마다 5천원의 회비를 납부하여야 하는데 신입회원에 한하여 2천원의 입회비를 부가하고 있다.

동 써클은 신입회원을 위하여 기초반을 두고 파스칼, MS-DOS를 매주 1회

의 모임을 통하여 교육하고 있으며, 정회원들은 3인 이상이 모여 SIG(Special Interest Group)라는 학술적인 모임을 구성하여 파스칼, 자료구조, 데이터베이스, 프롤로그 등을 공부하고 있다. 특히 방학중에는 15일동안 집중적인 교육을 실시하여 실력향상을 도모하고 있으며, 매년 1회의 전시회를 개최하여 많은 작품을 선보여 자신들의 실력을 평가받고 있고, 회지 '심지'의 활발한 발간을 통하여 써클의 단합을 유지하고 있다. 회원들이 자비로 출간하고 있는 심지는 현재 22호까지 발간되었으며 다른 학교 학생들이 서로 돌려가며 읽을 정도로 인기가 높다.

동 써클의 특징은 철저한 회원제와 사부제도인데 준회원이 정회원으로 승격됨에 있어 소정의 교육과정을 거치면서 학술부에서 제시한 프로젝트를 완성하여 제출한 후 심사에 합격하면 정회원으로 인정하고 있다. 특별회원은 졸업생과 S.C.S.C의 발전에 공헌한 사람이라고 인정된 사람이다.

사부제도는 한명의 사부와 3~5명의 신입회원으로 구성되어 신입회원의 써클적응, 학술 교육강화와 정회원 배출 등을 위하여 마련되었으며, 비교적 활발히 운영되고 있다.

김사흠 회장(공업화학과 3)은 학원문화와 관련된 세미나 및 정회원의 보다 많은 배출을 위해 기초반을 강화하면서 신입회원의 활발한 전시회 참가를 권장하기 위해 학술적인 면에 중점을 두고 89년 계획을 진행시킬 것이라고 한다.

S.C.S.C에 관한 문의사항은 전화번호 (02)599-1304 김사흠 회장 앞으로 하면 된다.

●연세 치대 컴퓨터반●

동계방학을 이용한 교육으로 실력 비축

연세대학교 치과대학 컴퓨터반은 86년 3월에 개인용 컴퓨터의 이해와 사용으로 생활에 도움이 되게 하고, 치의학 분야에서 그 활용도의 증진과 지도교수 및 회원 상호간의 친목도모를 목적으로 창립되었다.



20일간의 동계 트레이닝후 골수분자들만 모여서 찰칵

동 써클은 86년부터 자체 실력을 향상시키기 위해 매년 전시회를 주최하여 왔고, 특히 타 의과대학의 컴퓨터 써클이 소프트웨어에 많은 비중을 두고 있는 반면에 하드웨어도 공부하고 있다.

회원의 교육은 예과생과 신입생을 위하여는 2개의 그룹으로 나누어 MS-DOS와 파스칼을 매주 1회의 모임을 통하여 실시하고 있으며, 본과생을 위하여는 2주에 1회의 모임을 통하여 파스칼과 유틸리티, 하드웨어 세미나를 실시하고 있다. 그리고 지난 1월 9일부터 2월 3일까지 동계방학중에 20일간 워드스타, 파스칼, 로터스, 디베이스Ⅲ+ 등을 집중적으로 교육하는 등 실력향상에 주력하고 있다.

현재 회원은 본과생 22명, 예과생 12명으로 구성되어 있으며 신입생은 특별한 기간을 두지 않고 예과생에 한하여 모집하고 있으며, 회비는 매학기 1만원을 납부해야 한다.

신경진 회장(본과 2)은 89년도에는 무엇보다도 써클의 전통을 세우는데 중점을 두고 계획을 진행시킬 것이라고 한다. 특히 전시회에 보다 많은 회원들이 새로운 창작품으로 참여할 수 있도록 학술적인 면의 교육을 강화하며, 치과 원무에 관련된 프로그램의 개발에도 힘쓸 것이라고 한다.

연세 치대 컴퓨터반에 관하여 문의할 사항은 전화번호 (02)392-0131 치대 컴퓨터반으로 연락하면 된다.

●한국컴퓨터 선교회●

컴퓨터를 이용한 해외 선교활동 추진

올바른 컴퓨터의 활용과 복음을 전파한다는 취지아래 1986년 12월에 한국컴퓨터 선교회가 세워졌다.

동 선교회는 타 선교회와는 달리 선교방법이 컴퓨터라는 첨단기기를 통하여 전한다는 것이 다른 점이다. 이것은 급속도로 발전하고 있는 과학기술을 사용함에 있어, 특히 컴퓨터 사용분야에서 발생되는 범죄를 선교를 통하여 예방하기 위해 선교회에 컴퓨터를 설치하여 회원 누구나 사용할 수 있도록 개방하고 있고 컴퓨터 교육도 실시하고 있다. 그리고 교적관리, 목회자료관리, 교육자료관리 등 교회업무에 관한 프로

그램을 개발하여 교회 전산화를 이룩하는데 자문 및 지원을 하고 있다.

현재 회원이 전국적으로 300여명에 이르고 있으며, 대학생, 신학생, 회사원, 목사, 교수 등 다양한 직업의 회원으로 구성되어 있다. 회원은 입회원서와 1만원의 등록비를 납부해야 하는 등록회원, 등록회원이 된 후 6개월간 일정한 회비와 기도 및 활발한 활동을 보인 준회원, 준회원이 된 후 6개월간 일정한 회비와 기도 및 활발한 활동을 보인 정회원으로 구분되어 있다. 이들이 납부하고 있는 회비는 회원관리를 위하여 사용되고 있다.

동 선교회는 1명의 선교사와 회장, 부회장, 총무, 2인의 서기, 2인의 회계, 전도부, 교회정보부, 소프트웨어 개발부, 교육부, 문서선교부, 해외 선교부 등의 각부 부장이 주축이 되어 제반운영을 담당하고 있다. 매주 목요일에 정기예배를 실시하고 있는 동 선교회는 가입을 원하는 사람은 누구나 문을 두드릴 수 있다고 한다.

이영제 선교사(33)는 89년도 계획으로 자신들의 실력향상 및 홍보활동의 하나인 전시회를 종합전시장에서 개최하기 위해 노력할 것이며, 기독교와 컴퓨터가 관련된 세미나를 전시회와 병행하여 개최할 것이라고 한다. 그리고 장기적으로는 성경의 완벽한 전산화를 위한 계속적인 개발과 중동지역의 컴퓨터를 이용한 선교활동을 추진할 것이라고 한다.

동 선교회에 관한 문의사항은 전화번호 (02)676—7721 한국컴퓨터 선교회로 하면 된다.

● Chips ●

컴퓨터 대중화를 힘쓸터

컴퓨터의 칩처럼 구성원 개개인이 모두 자신의 역할을 다한다는 의미의 명칭으로 부산, 영남지역의 컴퓨터 써클 연합회(Chips)가 지난 85년 11월 부산지역 5개 대학의 컴퓨터 써클이 주축이 되어 설립되었다.

부산 및 영남지역 컴퓨터 문화의 대중화와 올바른 인식, 회원교 상호간의 유대강화와 정보교환을 목적으로 두고 활동하고 있는 동 연합회는 현재 경성대, 동아대, 동의대, 동주여자전문대, 부산대, 부산공대, 부산수산대, 부산외국어대, 울산대 등 9개 대학의 1500여명의 회원으로 구성되어 있다.

동 연합회는 각 회원교에 2명의 운영위원을 두고 회의 모든 사항을 운영위원회의 결정에 따라 처리하고 있으며, 운영위원회내에는 회장과 총무가 있어 Chips를 대표하고 결정사항을 집행하고 있다.

동 연합회는 회원교간의 정보교환과 전체적인 회원교육을 위하여 비정기적인 세미나와 학술회를 실시하고 있으며, 개별적인 학교단위의 교육은 회원교 자체에 일임하고 있고, 전회원교간의 친목도모를 위하여는 체육대회를 실시하고 있다. 그리고 매년 1회의 전시회를 개최하여 회원들의 창작품을 전시함과 동시에 일반 시민들에게 컴퓨터에 관한 상담을 실시하여 부산과 영남지역의 컴퓨터 인구 확대를 위한 발판을 마련하고 있다.

류병욱 회장(부산수산대 응용물리 3)은 우선 곧 마련될 Chips의 사무실을 중심으로 모든 사업을 실시할 예정이며, 특히 회원교의 확대와 학술발표회를 강화하고 향상된 전시회 개최를 위한 준비에 만전을 기할 것이라고 한다.

동 연합회에 가입을 하려면 부산, 영남지역에 있는 대학의 컴퓨터 써클로 동 연합회와 목적을 같이하는 단체이면 되며, 회원에 가입되면 매학기당 1만원의 회비를 납부해야 한다. 동 연합회에 대하여 알고 싶은 사항은 전화번호 (051)255—1646 류병욱 회장 앞으로 하면 된다.

알려드립니다.

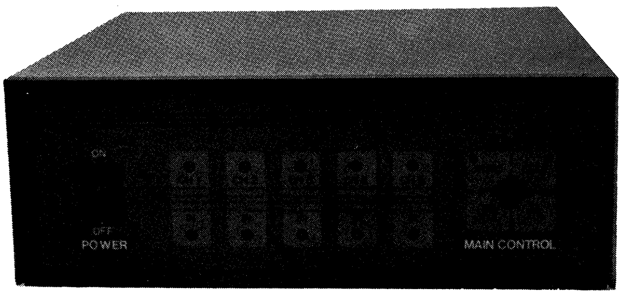
본지에서는 써클동정을 통해 전국에 산재되어
있는 아마추어들의 모임을 소개하여
동호인끼리 컴퓨터 이용자 모임을 형성할 수
있도록 유도하고 있읍니다.
이용자 모임을 알리고자 하는
분은 본지로 연락해
주시기 바랍니다.

연 락 처 : 서울시 용산구 갈월동 17번지
세기빌딩 ㊞ 140-150
전화번호 : (02)754—1044
F A X : (02)757—1049

PRINTER 주변기기 전문 메이커

- 컴퓨터는 2대 혹은 3대이고, 프린터는 1대뿐이라 불편하게 사용하고 계십니까?
- 컴퓨터 1대에 프린터를 1대씩 사용하거나 추가 구입시 프린터를 구매하려고 생각하십니까?
- 프린터를 하고 있는 동안 컴퓨터를 장시간 사용하지 못하고 계십니까?
- CAD 사용시 Plotter가 그리고 있는 동안 컴퓨터를 장시간 사용하지 못하고 계십니까?
- 컴퓨터의 수량과 프린터의 수량이 안맞아 적절한 사용을 못하고 계십니까?
- 학교 및 학원의 실습실에 프린터대수가 부족해 적절한 교육을 못하고 계십니까?
- F.A 혹은 계측기쪽의 프린터 인터페이스에 문제가 있읍니까?

컴퓨터 사용중에 불편한 위의 모든 문제를 자신있게 처리해 드리므로, 귀사의 업무에 도움이 되어 드리고 싶읍니다!



※ K-P-345A는 컴퓨터 프로그램 보호법에 의해 등록된 제품입니다.

K-P-345A의 사양

입·출력 방식	Centronics Type
입력 COMPUTER	3 : 1~5 : 1
출력 PRINTER	1 대
대기 시간	5초, 10초, 20초, 40초 중 선택
채널 변환방식	전자식
채널 선택방식	자동 or 반자동
용지 F.F 기능	자동 F.F or None F.F
F.F 삭제 기능	자동 삭제 or None

K-256P의 사양

버퍼 용량	256kB(80col 약 200장)
입·출력 방식	Centronics Type
입·출력 수	1 : 1 or 2 : 1
사용자 MODE	10EA
고유 기능	Data 중 Space 입력 기능, 재 Copy 기능, 메모리 기능, Self Test 기능 등 총 10가지



DATA SWITCH

- 전자식 완전자동 (2 : 1~8 : 1 or 64 : 1)
- 전자식 반자동 (2 : 1~8 : 1, 4 : 2)
- Serial 완전자동(4 : 1, 8 : 1)

DATA BUFFER

- 소용량 64kB
- 고성능 다기능 256kB
- 대용량 다기능 1MB
- Serial용 1MB

DATA CONVERTOR

- Serial→PARALLEL
- PARALLEL→Serial

각급 학교 및 학원의 전산 실습실 프린터 공유장치 공사

ATTA SYSTEM
PRINTER 주변기기 전문메이커
서울·종로구 종로 3가 175-4(현대APT709호)
전 화 : 2 7 7 - 1 7 3 9

● 상기 제품은 아래의 전문 취급점에서 전시 판매하고 있으며 직접 사용해 보실 수 있읍니다. ●

서울 : 세운상가 협진 274—5419(가동 라열 426호)
용산 나진상가 동해전자 719—8010(17동 가열 304호)
용산 전자랜드 협진 712—2764(3층 C-336호)

인천 : 바이너 리컴퓨터 (032) 762-5544
대구 : 범진 컴퓨터 (053) 252-2211
마산 : 한길 컴퓨터 (0551) 94-5555
원주 : 중부 시스템 (0371) 43-7979

(기타 지역 전문 취급점 모집중)

조건문

박성현 /디베이스 프로그래머

데이터의 출력

디베이스에서는 데이터를 어떻게 입력하고 정리하는가 하는 것도 중요하지만 그 결과를 출력해 내는 것도 중요하다. 이번호에서는 데이터를 출력해 내는 명령에 관하여 살펴보자.

우선 데이터를 화면에 출력해 내는 명령으로는

```
? / ??
@ yy,xx SAY
```

등이 있으며, 데이터를 프린터에 출력해 내는 명령은

```
... TO PRINT
SET PRINT ON/OFF
SET DEVICE TO SCREEN/PRINTER
```

등이 있다.

●? / ??

디베이스에서 이 명령은 물음표 그대로 "무엇이냐?"하는 명령이다.

```
. ? 20+50
      70
. var1=50
. ? 20+var1
      70
. ? var1
      50
```

앞의 명령과 같이 "20 더하기 50이 무엇이냐?"하고 물으면 70이라고 답을 나타낸다. 그리고 varl에 50이라는 숫자를 지정해 놓은 후 20 더하기 varl이라는 변수값은 얼마인지 물어본다. 역시 70이라고 답을 표시한다. 이번에는 varl이라는 변수가 무엇인지 물어 보면 50이라고 답한다.

똑같은 방법으로 이번에는 변수 대신 직접 그 내용을 물어 보자.

```
. ? "박 성 현"
박 성 현
. ?100
100
```

이와 같이 물어 본대로 한번에 한줄의 텍스트 내용이 화면에 나타난다.

여기서 답이 나오는 상태를 볼 때 '?'는 출력이 되기 전에 줄바꾸기가 생기고 '??'는 줄바꾸기를 하지 않은 채 출력을 한다. 그러므로 차례로 프린트를 하다 보면 맨 마지막 줄은 항상 남아 있게 된다. 이때는 '?' 명령을 한번 더 해주어야 할 것이다.

●@ yy, xx SAY…

'?'는 프롬프트가 있는 그 위치에서 스크롤되어 올라가면서 데이터를 출력해 내지만 이 명령은 화면상의 특수한 위치인 x, y 좌표를 지정해서 SAY 명령으로 표시된 메시지나 변수명을 출력하도록 한다.

화면의 좌측 맨 위의 좌표가 0, 0이고 화면의 우측 맨 아래의 좌표는 24, 79인 가로 80줄, 세로 25줄의 모니터에서 출력하고자 하는 첫글자의 위치를 지정한다.

```
. @ 10,10 SAY "박 성 현"
. @ 15,30 SAY var1
```

이렇게 매우 간단히 원하는 변수나 필드 내용을 읽어내서 볼 수 있다. 이와 같이 디베이스에서는 간단히 화면에서 "무엇이냐"하고 물어 보는 방법과 좌표를 지정하여 그곳에 "말해라" 하는 방법으로 출력을 얻어낸다. 이외에 지금까지 배운 LIST, DISPLAY 등도 출력을 얻어내는 명령의 일부라 할 수 있다.

이제 입력된 내용을 프린트해서 보는 방법으로 앞에서 소개한 3가지 방법을 살펴보자.

●…TO PRINT

이 명령은 지금까지 공부해서 알고 있는 명령이나 화면에 나오도록 하는 대부분의 명령의 뒤에다 TO PRINT 라는 옵션을 추가함으로써 화면에 출력함과 동시에 프린터에 출력한다.

```
. LIST <필드이름> TO PRINT
. LIST STATUS TO PRINT
. DISPLAY TO PRINT
. TYPE <화일이름> TO PRINT
```

●SET PRINT ON / OFF

이 명령은 MS-DOS에서 Ctrl-P를 누른 효과와 거의 같은 명령이다.

```
. SET PRINT ON
```

이라고 명령을 하면 그 다음부터는 화면에 나오는 모든 것이 프린터로 동시에 출력된다. 단, 화면의 좌표를 지정하여 출력이 나오는 명령의 결과치는 나오지 않는다. 그러므로 @ yy, xx SAY … 명령의 결과치는 화면에만

나오고 프린터에는 나오지 않는다. 그리고

```
. SET PRINT OFF
```

하게 되면 프린터 출력이 정지된다.

●SET DEVICE TO SCREEN / PRINTER

이 명령은 @ yy, xx SAY… 명령을 화면에 출력시킬 것인가 또는 프린터로 출력시킬 것인가를 지정해 주는 명령이다. 만약 SET DEVICE TO PRINTER 라고 명령을 주면 y, x 좌표가 0, 0에서 24, 79까지만 나오는 것이 아니라 프린터의 가로줄 최대치까지 사용할 수 있고, 세로는 무한정 사용할 수가 있다.

만약 세로의 크기가 방금전에 사용했던 것보다 작은 숫자를 사용하면 한 페이지를 넘긴 후 그다음 세로줄의 위치에 프린트하게 된다. (라인 프린터는 거꾸로 움직일 수 없으므로 그 위치에 프린트하기 위해서는 다음 페이지에 프린트하게 된다.)

```
. SET DEVICE TO PRINTER
```

라고 명령한 후의 @ yy, xx SAY… 명령은 화면에는 나오지 않고 프린터에만 나오게 된다.

다음은 프린터와 관계되는 명령을 알아 보자.

●EJECT

이 명령은 아스키코드 12번 값을 프린터로 보내는 것으로 프린터의 용지를 다음 페이지 처음으로 옮기는 명령이다.

```
. EJECT
```

라고 명령하면 종이 한장을 그대로 넘긴다.

●SET MARGIN TO N

이 명령은 왼쪽 여백을 바꾸어 줄 때 사용된다. 즉, 프린터의 첫번째 세로 위치를 지정하게 한다.

```
. SET MARGIN TO 20
```

라고 지정한 다음부터는 0, 20의 위치가 0, 0으로 지정되어 항상 세로칸 20개를 띄운채 프린트하게 된다.

●프린트 제어

프린터에서 한글모드로 바꾸어야 할 경우에는 앞에서 설명한 2가지의 명령으로 프린터를 제어할 수 있다. (엡슨 프린터의 제어 코드로 예를 들어 보겠다) (화면 1, 2 참조)

디베이스에서의 프로그래밍

디베이스가 이토록 인기가 있고 그 유용성을 인정받는 것은 디베이스가 지니고 있는 자체의 프로그래밍 능력이라 하겠다. 디베이스에서의 프로그래밍 언어는 어떠한 것인가 다음의 명령을 해보면 간단히 알 수 있다.

```
. modify command test
```

라 명령하면 디베이스 자체의 워드프로세서가 나타날 것이다. (만약 config. db에 tedit=〈워드프로세서 화일이름〉을 주었다면 지정한 워드프로세서가 나온다.)

여기서는 워드프로세서 사용법의 설명은 생략하기로 한다. 워드프로세서로 지금까지 배운 명령을 몇가지 입력해 보기로 하자.

```
? 1000+200
list status
dir
var="I love programming"
@ 10,10 say var
wait
clear
use pc01
list
```

이와 같이 입력하면 test.prg라는 화일이 생성된다. 여기서 .prg라는 확장자를 붙이지 않더라도 디베이스에서는 자동으로 붙여 준다.

다음은 이 화일을 실행시켜 보자.

```
. do test
```

〔화면 1〕

```
SET PRINT ON
? CHR(27)+'h'+CHR(27)+'W' <--- 한글모드 지정, 영한 비율 2:1로 지정
? CHR(27)+'B'+'박 성 현' <--- 이중 인쇄로 지정
  :
  :
  :
SET PRINT OFF
```

〔화면 2〕

```
SET DEVICE TO PRINTER
@ 0,0 SAY CHR(27)+'h'+CHR(27)+'W' <--- 한글모드 , 영한 비율 2:1로 지정
@ 0,0 SAY CHR(27)+'D'+'박 성 현' <--- 확대문자 지정
  :
  :
  :
SET DEVICE TO SCREEN
```

어떠한 결과가 일어나는가? 도트 프롬프트에서 입력한 것과 마찬가지로 한꺼번에 실행됨을 볼 수 있다.

이와 같이 디베이스에서의 프로그램이란 도트 프롬프트에서의 명령을 한꺼번에 모아 놓은 것과 같음을 알 수 있다. (마치 도스에서의 명령을 묶어서 한꺼번에 처리하는 배치화일과 같이)

여기서 프로그램의 기본 구성을 위한 명령으로 아래와 같이 Corado Bohm & Guiseppe Jacopini의 구조화 프로그램 기능이 포함되어 있다(그림 1 참조).

① 순차제어구조(Sequence)

순자제어구조는 하나의 행위가 끝나면 다음 행위가 바로 시작되는 동작을 말한다(그림 1의 (a))

② 조건제어구조(If-then-else)

IF 다음에 나오는 조건이 True인지 False인지 비교하여 True이면 다음의 명령을 수행하고 False이면 건너 뛰어서 ELSE 이후의 명령을 수행한다. 만약 ELSE가 없다면 ENDIF 이후의 명령을 수행한다(그림 1의 (b)).

디베이스에서는 IF문은 단지 한번의 비교를 하지만 DO CASE는 여러개를 한꺼번에 비교하므로 단순한 구조로 되어 알아보기 쉬울 뿐만 아니라 IF문의 수를 줄이는 결과가 되므로 더 빠른 처리를 할 수 있는 잇점도 있다(그림 2 참조).

③ 반복 제어구조(Do while)

순환(LOOP) 처리를 사용하는 데 쓰이며 조건이 만족할 때까지 계속해서 수행한다(그림 1의 (c)).

디베이스에서는

```
DO WHILE < 조건 >
 ........
 ..........
ENDDO
```

DO WHILE과 ENDDO 사이에 'LOOP' 와 'EXIT'라는 명령을 수행하여 각각 순환의 처음으로 돌아가게 하거나 (LOOP), 순환에서 빠져나오게 하는 명령(EXIT)도 제공된다.

이런 구조화 프로그램에서 가장 중요한 것은 3가지 제어구조 각각에는 반드시 하나의 ENTRY POINT(입구)와 EXIT POINT(출구)가 있다는 점이다. 다시 말해 **그림 3**과 같이 표현할 수 있다.

그러면 루프를 사용하여 1부터 100까지의 합계를 구하는 프로그램을 작성하여 보자.

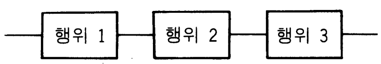
우선 프로그램을 코딩하기 전에 프로그램을 구조적으로 설계하기 위해 Pseudo code를 작성해 보자.

```
RESULT 를 0 로 지정
DATA를 1 로 지정
DO WHILE DATA가 100 보다 작거나 같은 동안
        RESULT에 DATA를 더한다
        DATA를 1 증가 시킨다.
ENDDO
```

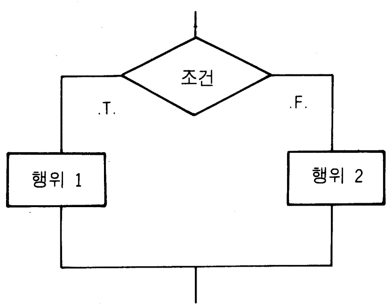
이제는 프로그램을 작성해 보자.

```
SET TALK OFF
result=0
data=1
DO WHILE data <= 100
        result = result + data
        data = data + 1
ENDDO
? result
SET TALK ON
```

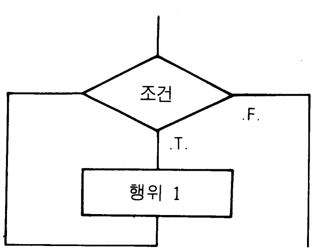
〔그림 1〕



(a) 순차제어 구조

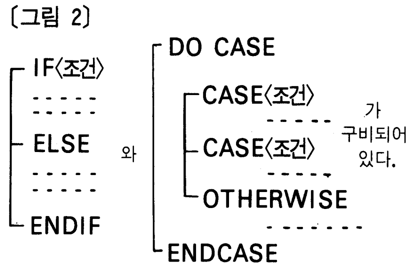


(b) 조건제어 구조

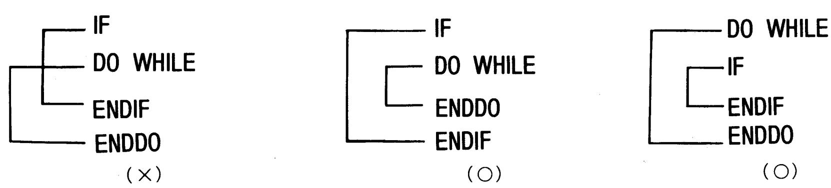


(c) 반복제어 구조

〔그림 2〕



〔그림 3〕



● **Pseudo code**

Pseudo code라 함은 흐름도(Flow chart)의 대체안으로 프로그램의 논리를 영문으로 쓰는 논리 단계를 말하는 것으로써 여러가지 장단점이 있지만 dBASE Programming Languge는 Pseudo code와 흡사하여 흐름도보다는 Pseudo code를 사용함이 편리하다.

여기서 SET TALK OFF는 도트 프롬프트에서의 작업처럼 대화식모드를 비대화식 모드로 바꾸기 위한 명령이다. 만약 SET TALK ON 상태로 프로그램을 실행한다면

```
result=0
```

라고 정의했을 때 디베이스에서는 "그래 0이다"하는 식으로 0이라고 화면에 표시를 해준다. 프로그램을 수행할 때에는 이와 같이 비대화식 모드로 전환하여 불필요한 응답을 없애야 한다.

다음은 디베이스 화일을 이용한 프로그램을 작성해 보자.

우선 디베이스 화일의 구조를 만든 다음, 필드를 지정해 줄 때 다음과 같이 한다.

```
.create calc
        NAME -- Character --- 10
        PAY  -- Numeric ---- 7
```

이렇게 지정하였으면 화면 3같이 DO WHILE 구조를 이용하여 작성한다.

EOF()라는 함수는 현재는 레코드 위치가 마지막인가 아닌가를 체크해주는 함수로써 만약 현재의 레코드가 마지막이라면 .T.가 되지만 앞의 .NOT.의 영향으로 조건 전체는 결국 .F.가 된다. 따라서 DO WHILE 내에서 빠져 나오게 된다. 만약 DO WHILE .T.라고 명령한다면 항상 .T.이므로 컴퓨터를 끄지 않는한 계속 DO WHILE--- ENDDO를 수행할 것이다.

앞의 프로그램은 화면 4와 같이 바꿀 수도 있다. 다음은 데이터를 찾는 프로그램을 IF와 DO CASE문을 이용하여 작성해 보자(화면 5, 6 참조).

DO CASE에서는 여러개의 CASE 중에서 단 1번만 선택되어 실행된다. 예를 들면 PAY가 29만원이어서 1.1배를 한후 다시 31만 9천원이 되었다 해도 다음 CASE에서 30만원 이하 50만원 미만에 해당이 되어 다시 실행되는 것이 아니고 곧바로 CASE 구조에서 빠져 나오게 된다. PC

〔화면 3〕

```
USE CALC                              && CALC.DBF 화일을 연다.
DO WHILE .NOT. EOF()                  && 화일의 끝이 나타날 때까지 계속
      ? "NAME : ",NAME,"PAY: ",PAY && 이름과 급여를 디스플레이
      SKIP                            && 다음 레코드로 이동
ENDDO
```

〔화면 4〕

```
USE CALC                              && CALC.DBF 화일을 연다.
DO WHILE .T.                          && 영원히 계속
      ? "NAME : ",NAME,"PAY: ",PAY && 이름과 급여를 디스플레이
      SKIP                            && 다음 레코드로 이동
      IF EOF()                        && 화일 끝이면
            EXIT                      && do while 밖으로 빠져 나간다.
      ENDIF
ENDDO
```

〔화면 5〕

```
USE CALC                              && CALC.DBF 화일을 연다.
mNAME=SPACE(10)                       && MNAME 이라는 변수의 초기치를 지정
@ 10,10 SAY "NAME : " GET mNAME       && 받아들일 변수 지정.
READ                                  && 변수를 받아 들인다.
LOCATE FOR NAME=mNAME                 && 필드에서 변수와 같은 것을 찾는다.
IF FOUND()                            && 만약 찾았다면
      @ 12,10 SAY "PAY : "            && 급여 내용을 디스플레이
      @ 12,17 SAY PAY
ELSE
      @ 12,10 SAY " Not Found !" && 못찾았다면 못찾았다는 안내
ENDIF
```

〔화면 6〕

```
USE CALC
DO WHILE .NOT. EOF()                                  && 화일의 끝이 아닌 동안 계속
      DO CASE
            CASE PAY < 300000                         && PAY 가 30 만원보다 적을 경우
                  REPLACE PAY WITH PAY*1.10
            CASE PAY >= 300000 .AND. PAY < 500000 && PAY 가 30 만원이상 50 만원 미만인 경우
                  REPLACE PAY WITH PAY*1.08
            CASE PAY >= 500000 .AND. PAY < 700000 && PAY 가 50 만원이상 70 만원 미만인 경우
                  REPLACE PAY WITH PAY*1.06
            CASE PAY >= 700000 .AND. PAY < 900000 && PAY 가 70 만원이상 90 만원 미만인 경우
                  REPLACE PAY WITH PAY*1.04
            OTHERWISE                                 && 위에 해당 되지 않은 모든 경우
                  REPLACE PAY WITH PAY*1.02
      ENDCASE
      SKIP                                            && 다음 레코드로 이동
ENDDO
```

프로그램내에서의 데이터 관리

박승제/여주컴퓨터학원장

지난 호에서 우리는 어셈블을 하는 과정과 1문자 출력, 그리고 1문자를 입력하는 방법에 대해서 배웠다. 이번에는 프로그램속에서 어떻게 데이터를 설정하는지에 대해서 알아보자.

데이터 세그먼트의 설정

프로그램수행중에 필요한 데이터를 두는 방법은 다음과 같이 2가지가 있다. 첫번째 방법은 데이터를 프로그램의 일부로 작성하는 것이고, 두번째 방법은 데이터만을 모아 별개의 화일로 작성하여 디스크에 수록한 후 필요시 프로그램을 실행하면서 로드하는 방법이다.

이들 중 두번째 방법은 데이터가 대량일 경우에만 사용하며 보통은 데이터를 프로그램 코드내부에 포함시켜 버린다. 왜냐하면 프로그램은 반드시 메모리에 로드하여 실행하게 되므로 프로그램이 로드되면 자연히 데이터도 로드되어 따로 데이터를 로드하기 위한 루틴을 설계하지 않아도 되기 때문이다.

그리고 프로그램상에 데이터를 둘때는 프로그램 코드와 구별하기 위해서 프로그램의 맨 끝부분에 위치시키는 것이 일반적이다.

데이터를 프로그램상에 두는 경우에는 먼저 '데이터 세그먼트 레지스터(DS)'의 값을 설정해 주어야 하는데 이는 8088 CPU가 메모리에 저장된 데이터를 가져올 때는 그 데이터가 데이터 세그먼트에 있다고 가정하기 때문이다 (그림 1 참조).

프로그램의 길이가 짧은 경우에는 보통 DS의 값을 '코드 세그먼트 레지스터 (CS)'와 일치시켜 버린다. DS를 설정하는 과정은 다음과 같다.

```
MAIN   SEGMENT
       ASSUME   CS:MAIN, DS:MAIN
       MOV      AX,CS
       MOV      DS,AX
        :        :
```

'MAIN'은 세그먼트 이름으로 사용자가 임의로 지정한다는 것은 이미 지난 호에서 설명한 바 있다. 이것은 'CODE' 도 좋고 'LIST'라도 좋다. 사용자 단어의 조건에만 맞는다면 아무 것이라도 괜찮다.

그 다음의 ASSUME문은 CS의 값을 프로그램의 첫머리와 일치시키는 과정이다. CS 레지스터는 항상 프로그램의 시작번지를 기억하게 되므로 이 의사명령만 기술해도 되지만 DS는 이것만으로는 초기설정이 되지 않는다. 그러므로 CS 값을 읽어내서 DS에 전송하는 방법으로 두 세그먼트 레지스터의 값을 일치시킨다.

이때 'MOV DS, CS'처럼 CS의 값을 DS로 직접 이동하려 하면 에러가 발생하게 된다. 8088에는 세그먼트 레지스터끼리 데이터를 전송하는 명령이 없으므로 AX 레지스터를 통해 전달시키고 있는 것이다. 위의 방법은 어셈블리에서는 정석과 같은 것이므로 반드시 외워두어야 한다.

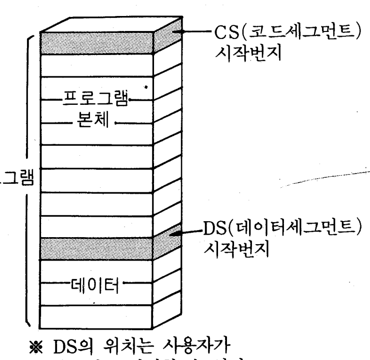
메모리에 저장되어 있는 데이터는 'MOV'명령으로 레지스터로 옮길 수 있다. 이때는 '메모리 몇 번지에 있는 데이터를 어느 레지스터로 옮겨라'라는 식으로 명령을 내리게 되는데 이것은 어셈블리 명령으로 표현하면 다음과 같다.

MOV 레지스터, 메모리 번지

여기서 '메모리 번지'는 그 번지에 들어있는 값이라는 의미이다. 이렇게 번지를 지정할 경우에는 프로그래머가 데이터가 놓여지는 번지를 기억하고 있지않으면 안된다. 그런데 위에서 데이터는 프로그램의 끝부분에 위치시킨다고 하였으므로 프로그램의 코드 크기가 바뀌면 당연히 데이터의 위치도 바뀌게 된다. 또한 이 번지를 프로그래머가 일일이 계산하여 지정하는 것은 여간 큰 일이 아닐 수 없다.

그래서 어셈블리 언어에서는 의사명령을 이용하여 데이터를 설정하고, 데이터가 위치하는 번지는 어셈블러가

〔그림 1〕 데이터 세그먼트



※ DS의 위치는 사용자가 마음대로 지정할 수 있다.

계산하여 부여하도록 하는 방법을 이용한다. 이렇게 되면 프로그래머는 의사 명령만 잘 다룰줄 알면 메모리번지는 어셈블러가 부여하므로 번지 등에는 전혀 신경쓸 필요가 없어진다. 메모리에 데이터를 설정하는 경우에 쓰이는 의사명령은 다음과 같다.

DB : 1바이트의 문자, 수치
DW : 워드(2바이트)
DD : 더블 워드(4바이트)
DQ : 쿼드 워드(8바이트)
DT : 10바이트

한편 의사명령의 표현형식은

레이블 DB 식

과 같으며 레이블은 사용자 단어의 규칙에 어긋나지 않는다면 어떤 문자열이라도 좋다. (이 규칙은 지난 호에 설명했던 첫문자는 영문자, 그 외는 영문자나 숫자, '_'로 이루어진 31문자 이내의 문자열을 말한다.)

이렇게 설정하고 MOV AL, '레이블'등과 같이 지정하면 AL레지스터에 레이블이 가르키는 번지의 값이 레지스터로 이동하게 된다. 즉 베이직의 변수와 같은 개념으로 이 레이블을 활용할 수 있다. 그러므로 이 레이블을 '변수'로 표기하기도 한다. 예를 들어 메모리의 어딘가에 '1234H' 라는 값을 저장했다가 불러오는 과정을 어셈블리 코드로 표현하면 다음과 같다.

```
MOV   AX, xxx
 .     .
 .     .
xxx   DW  1234H
```

이때 불러들이는 레지스터의 크기와 저장된 변수의 크기가 같아야 함은 물론이다. 그러면 위의 내용을 실 예를 통해 익혀 보기로 하자.

[예제 1]
메모리에 'ABCDE'를 저장한 후 1문자씩 출력한다.

리스트 1의 프로그램을 보면 DATA1, DATA2등과 같이 변수를 이용하여 프로그램상에 데이터를 설정하고 있으며 이것을 'MOV DL, DATA1'과 같이 읽어들이는 것을 볼 수 있다. 또한 0004 번지를 보면 '8A 16 0028 R'과 같이 기계어 코드 끝에 'R'이 붙어있는 것을 볼 수 있는데 이것은 상대번지 지정방식임을 나타내고 있다.

이 상대번지 지정방식은 어셈블할 당시에는 실제의 번지가 부여되지 않고, 링커가 재배치할 때에 부여하는 것이라고 알고 있으면 된다.

또 000C 번지에 'MOV BX, DATA2'가 있는데 이때는 메모리의 상위번지의 값이 BH에, 하위번지의 값이 BL에 저장된다. 실제 이 데이터가 메모리에 저장될 때는 상위바이트와 하위바이트가 바뀌어 저장되게 된다. 즉, 메모리에 저장될 때는 43H, 42H순으로 저장된다는 말이다.

이것은 회로가 간단해진다거나 속도가 빨라진다거나 하는 CPU제조상의 이유인데 우리는 '상하가 바뀌어 저장된다'고 알고만 있으면 된다. 모든 CPU가 다 그런 것은 아니고 80계열은 상하가 바뀌게 된다. 이렇게 저장하는 방법을 '역워드 방식'이라고 한다.

데이터는 문자형태로 기술해도 되고 수치형태로 기술해도 된다. 수치나 문자 모두 메모리에 저장될 때는 2진수 형태이므로 이 자료를 수치로 취급하느냐 문자로 취급하느냐는 처리하기에 달려있다.

〔리스트 1〕 'ABCDE'를 출력하는 프로그램

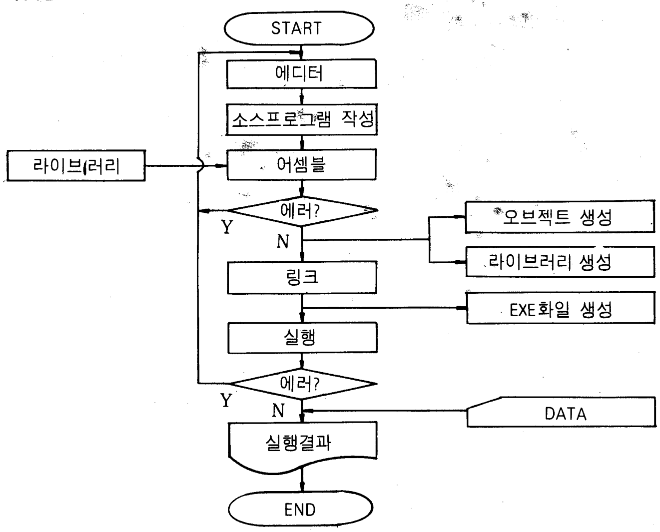
```
0000                       MAIN     SEGMENT
                                    ASSUME CS:MAIN,DS:MAIN
0000  8C C8                         MOV      AX,CS
0002  8E D8                         MOV      DS,AX
0004  8A 16 0028 R                  MOV      DL,DATA1
0008  B4 02                         MOV      AH,2
000A  CD 21                         INT      21H
                                    ;
000C  8B 1E 0029 R                  MOV      BX,DATA2
0010  8A D7                         MOV      DL,BH
0012  CD 21                         INT      21H
                                    ;
0014  8A D3                         MOV      DL,BL
0016  CD 21                         INT      21H
                                    ;
0018  8B 0E 002B R                  MOV      CX,DATA3
001C  8A D5                         MOV      DL,CH
001E  CD 21                         INT      21H
                                    ;
0020  8A D1                         MOV      DL,CL
0022  CD 21                         INT      21H
                                    ;
0024  B4 4C                         MOV      AH,4CH
0026  CD 21                         INT      21H
0028  41                   DATA1    DB       'A'
0029  4243                 DATA2    DW       4243H
002B  4445                 DATA3    DW       'DE'
002D                       MAIN     ENDS
                                    END
```

0029H 번지는 자료를 문자형태로 저장하는 보기이며 002BH 번지는 수치(16진수) 형태로 저장하는 보기이다. 수치인 경우나 문자인 경우 어셈블된 결과를 보면 모두 16진수로 바뀌어 있는 것을 볼 수 있다. 자료를 문자형태로 두는 경우에는 그 자료를 작은 따옴표(')로 묶어주면 된다.

리스트 1은 지난 호에서 배운 DOS 인터럽트 21H를 호출하고 있으므로 프로그램의 논리를 이해하는데 별로 어려운 내용은 없을 것이다. 워드(DW)로 저장된 내용은 상하로 나누어 한 바이트씩 출력하게 된다.

입력이 모두 끝났으면 위의 내용을 디스크에 수록하고 어셈블과 링크를 해야 한다. 그림 2는 어셈블과 링크하는 과정을 나타낸 것이다.

〔그림 2〕 어셈블하는 과정



램디스크와 배치화일을 이용한 어셈블

어셈블리를 포함한 대부분의 컴파일형 언어에서는 프로그램을 작성한 후에 컴파일이라는 단계와 링킹하는 과정을 거쳐야 하는데 이 과정은 초보자뿐 아니라 전문가들에게도 상당히 지루하고 번거로운 작업이다.

에러가 발생한다거나 프로그램을 조금이라도 변경할 경우에는 에디터 수준까지 거슬러 올라간후 컴파일과 링크작업의 과정을 반복해야 한다. 그래서 이 과정을 조금이라도 빠르게 처리하기 위해서 램디스크의 이용을 권하고 싶다.

램디스크란 메모리의 일부를 할당하여 마치 디스크처럼 사용하는 것을 말하는데 기계적으로 동작하는 플로피디스크보다 최소한 200배는 빠르게 동작한다. 그러므로 웬만한 프로그램이라면 저장(Save)을 하거나 로드(Load)를 하는데 1초도 걸리지 않는다.

즉 소스나 오브젝트 화일등을 램디스크에 수록하도록 하면 디스크의 입출력 시간을 대폭 단축시킬 수 있다는 말이다. 또한 플로피디스크에 수록하는 경우 나중에 불필요한 화일은 삭제해야 하지만 램디스크에 모든 것을 수록하도록 하고 필요한 부분만 플로피디스크로 백업을 받으면 디스켓에 화일들을 질서정연하게 기록시킬 수 있게 된다.

단, 램디스크에 수록된 자료는 전원이 꺼지면 자동적으로 소멸되므로 정전등의 불의의 사고에 대비하여 필요한 자료는 가끔씩 디스켓에 백업해 둘 필요가 있음을 알아두자.

램디스크는 드라이브 번호로 C:(하드디스크 사용자는 D:)를 배정받는다. 램디스크를 설치하는 과정은 다음과 같다. (DOS 버전 3.00이상에서)

① DOS디스크를 A:, 공디스크를 B:에 넣고 FORMAT B:/S로 포맷한다.

② DOS디스크에서 VDISK.SYS를 B:에 복사한다.

③ 매크로 어셈블러 디스켓을 A:에 넣고 다음의 화일을 B:에 복사한다.

```
MASM.EXE, LIB.EXE, LINK.EXE, CREF.EXE
```

④ 사이드킥(SideKick)이 있으면 이것을 A:에 넣고 SK.COM을 B:에 복사한다. 사이드킥이 없으면 EDLIN.COM이나 다른 에디터를 B:에 복사한다.

에디터는 크기가 작은 것을 로드하는데도 시간이 많이 걸린다.

⑤ 다음은 CONFIG.SYS를 만들어야 한다.

```
A>COPY CON B:CONFIG.SYS
DEVICE=VDISK.SYS 128 ⏎
(128KB를 램디스크로 할당한다.)
<Ctrl>-<Z> ⏎
```

⑥ 자동 실행화일을 만든다.

```
A>COPY CON AUTOEXEC.BAT
ECHO OFF
DATE
C:   (하드디스크의 사용자는 D:가 된다.)
PATH A:
SK (사이드킥이 없는 사람은 다른 에디터 이름으로 대치)
<Ctrl>-<Z>
```

⑦ 어셈블러를 실행할 배치화일을 만든다.

```
A>COPY B:ASM.BAT
  MASM %1;
  LINK %1;
  <Ctrl>-<Z>
```

이 디스크를 A:에 넣고 시스템을 재부트(Reset)시키면 에디터상태가 된

다. 여기서 프로그램을 작성하고 저장하면 그 프로그램은 C:(하드디스크 사용자는 D:)로 수록된다. 이제부터는 'ASM 화일명'만 입력해도 어셈블과 링크작업을 자동적으로 수행하게 된다.

사이드킥은 램상주 프로그램으로 한번 로드시킨 후 계속 이용할 수 있으며 DOS 상태에서 사이드킥을 부를때는 (〈Alt〉+〈Ctrl〉로 호출한다) 단 1초도 안 걸리며 프로그램을 편집한 마지막의 상태를 유지시켜 주므로 프로그램을 작성하기에는 최적의 에디터라고 할 수 있다. 사이드킥을 사용해보지 않는 독자는 한번 사용해 보기 바란다.

문자열의 출력

위에서 배운 방법으로는 긴 문자열을 출력하려면 1문자씩 출력해야 하므로 대단한 수고를 해야 할 것이다. 긴 문자열을 출력하는 경우에는 시스템 기능 호출의 9번 기능을 이용한다. 이때 출력하려는 문자열의 끝은 '$'로 끝나야 하며 '$' 자체는 화면에출력되지 않는다. '$'를 출력하려면 앞에서 배운 시스템 기능호출 2번 기능을 이용한다.

```
MOV  AH,9
MOV  DX,문자열의 시작번지
INT  21H
```

문자열을 메모리에 기억시키는 요령은 위에서 배운 DB등의 의사명령을 이용한다. 예를 들어 'ABCDEF'를 두는 경우라면 다음 방법으로 기술할 수 있다.

```
• DB  'A','B','C','D','E','F','$'
• DB  'ABCDEF$'
• DW  'BA','DC','EF','$'
  (상위와 하위가 바뀌는 것을 고려한다.)
• DD  41424344H,4546H
  (DD,DQ,DT에서는 상하가 바뀌지 않는다.)
```

위에서 DB일 경우에는 'ABCD..'처럼 문자열을 계속해서 기술할 수 있지만 그 외에는 2바이트씩 나누어서 기술하여야 한다. 단 수치형태로 기술할 경우에는 예외이다. 그러면 예제를 통해 문자열을 출력하는 방법을 익혀보자.

[예제 2]

화면 1과 같이 출력하는 프로그램을 작성하라.

예제 2에서는 그래픽문자를 출력하고 있는데 그래픽문자는 키보드에서 직접 입력이 불가능하므로 아스키 코드를 구하여 출력하도록 한다. 아스키 코드표를 보면 그래픽 문자가 다음과 같이 할당되어 있다.

문자	코드
╔	201
═	205
╗	187
║	186
╚	200
╝	188

위에서 설명한 방법으로 '╔═════╗'을 출력하려면 다음과 같이 상당히 길어진다.

```
DB 201,205,205,205, ... ,187,0DH,0AH
```

매크로 어셈블러는 값이 같은 자료를 반복해서 나열하는 의사명령이 있으므로 이것을 사용하면 프로그램을 손쉽게 작성할 수 있다. 위의 의사명령은 다음과 같이 대치될 수 있다.

```
DB 201
DB 30 DUP(205)
DB 187,0DH,0AH
```

DUP'는 ( )안의 자료를 앞의 수치만큼 나열하라는 의미이다. 그러므로 이 명령에 의해 메모리에는 '═'문자가 30개 늘어선 효과를 얻는다.

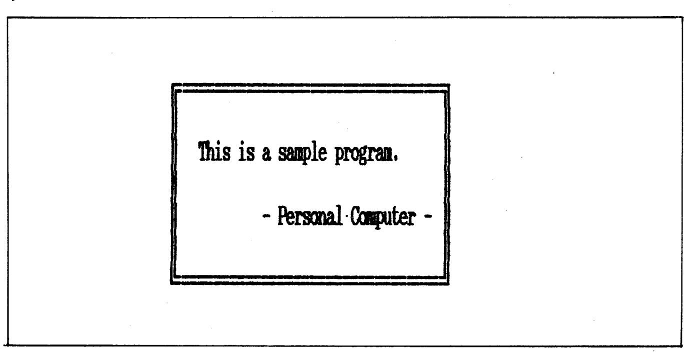
또 화면에 한줄을 출력한 후에는 줄을 바꿔야(Line Feed)하므로 화면상에서 줄을 바꿀 위치에는 0DH, 0AH의 2바이트가 수록되어 있어야 한다.

컴퓨터는 이들 제어문자도 하나의 캐릭터(문자)로 취급을 하며 위의 2바이트가 연속으로 나타나면 화면의 줄을 바꾸게 된다. 그러므로 이들을 하나의 문자열로 묶어 단 한번의 실행으로 위의 출력 결과를 얻을 수 있다. 리스트 2에 위의 문제의 해답을 수록했으니 실행해 보기 바란다.

리스트 2의 0006H를 보면 'MOV DX, OFFSET D1'처럼 기술되어 있다. 'MOV DX, D1'처럼 기술하면 'D1'이 위치하는 첫머리 번지의 데이터를 DX에 넣게 된다.

이 경우에 DX 레지스터는 16비트 크기이고 D1 다음의 자료는 'DB'처럼 1바이트 크기로 저장되므로 크기가 맞지 않아 에러를 발생하게 된다. 만일 에러가 발생하지 않는다 하더라도 이

〔화면 1〕



값을 직접 가져오면 문자열 전체를 출력할 수가 없게 된다. 왜냐하면 DOS 기능호출의 9번 기능을 이용하려면 '문자열이 저장된 번지'를 DX에 넣는 것이지 그 곳의 데이터 자체를 넣는 것이 아니기 때문이다.

거듭 말하지만 메모리의 번지와 번지내의 데이터라는 것은 초보자에게는 상당히 혼란스러운 개념이다. 이것은 비유적으로 표현하면 종로구 창신동 123번지에 홍길동이란 사람이 살고 있다고 할 때 홍길동의 주소(번지)는 123이고 이 123번지의 세대주(데이터)는 홍길동이라는 식이 된다(그림 3 참조).

어셈블리에서는 자료를 바이트 혹은 워드단위로 처리하는 경우에는 데이터(세대주)를 지정하지만 메모리에 연속되어 저장된 자료(창신동 전체주민)를 대상으로 작업할 경우에는 메모리의 주소(번지)를 지정하게 된다.

데이터가 저장된 번지를 지정할 경우에는 'OFFSET'라는 의사명령을 이용한다. 'MOV DX, OFFSET D1'이라는 명령에 의해 'D1'이란 레이블이 붙어 있는 곳의 번지(주소)를 DX에 넣게 된다.

이 'OFFSET'이라는 것은 '처음부터 세어서 몇번째 위치'라는 의미로 생각하기 바란다. 어셈블된 리스트를 보면 상대번지 기호인 'R'이 붙어 있는 것을 발견할 수 있다.

또한 'DUP'를 이용해서 자료를 반복 저장한 부분의 어셈블 된 결과를 보면 '[20]'처럼 되어 있는데 이것은 20H(10진수로 32:공백 문자)가 지정된 갯수 만큼 저장되어 있다는 것을 나타내는 것이다.

문자열의 입력

이번에는 문자열을 입력하는 것에 대해 알아보기로 하자. 문자열을 입력하려면 먼저 입력받을 문자열을 저장할 장소를 준비하여야 한다. 앞에서 소개한 문자열 출력의 경우에는 DS를

〔리스트 2〕 문자열을 출력하는 프로그램

```
0000                        CODE   SEGMENT
                                   ASSUME CS:CODE,DS:CODE
0000  8C C8                        MOV    AX,CS
0002  8E D8                        MOV    DS,AX
                                   ;
0004  B4 09                        MOV    AH,9
0006  BA 000F R                    MOV    DX,OFFSET D1
0009  CD 21                        INT    21H
000B  B4 4C                        MOV    AH,4CH
000D  CD 21                        INT    21H
000F  C9                    D1     DB     201
0010  001E[                        DB     30  DUP(205)
          CD
                     ]

002E  BB 0D 0A                     DB     187,0DH,0AH
0031  BA                           DB     186
0032  001E[                        DB     30  DUP(' ')
          20
                     ]

0050  BA 0D 0A                     DB     186,0DH,0AH
0053  BA 20 20 54 68 69 73         DB     186,' This is a sample program.

      20 69 73 20 61 20 73
      61 6D 70 6C 65 20 70
      72 6F 67 72 61 6D 2E
      20 20 20
0072  BA 0D 0A                     DB     186,0DH,0AH
0075  BA                           DB     186
0076  001E[                        DB     30  DUP(32)
          20
                     ]

0094  BA 0D 0A                     DB     186,0DH,0AH
0097  BA 20 20 20 20 20 20         DB     186,'- Personal Computer -

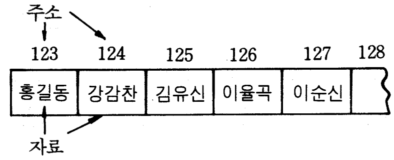
      20 2D 20 50 65 72 73
      6F 6E 61 6C 20 43 6F
      6D 70 75 74 65 72 20
      2D 20 20
00B6  BA 0D 0A                     DB     186,0DH,0AH
00B9  BA                           DB     186
00BA  001E[                        DB     30  DUP(32)
          20
                     ]

00D8  BA 0D 0A                     DB     186,0DH,0AH
00DB  C8                           DB     200
00DC  001E[                        DB     30  DUP(205)
          CD
```

```
00FA  BC 0D 0A 24                    DB     188,0DH,0AH,'$'

00FE                        CODE    ENDS
                                    END
```

〔그림 3〕 번지와 자료(데이터)의 개념



홍길동의 주소 : 123번지
123번지 내의 자료 : 홍길동
※ 번지를 지정하는데는 'OFFSET'이라는 의사명령을 기술한다(번지에는 통/반을 부여하지만 사람에게는 부여하지 않는다).

CS와 일치시키는 방법을 이용했지만 여기서는 DS를 독립적으로 설정하는 방법을 이용해 보기로 한다.

[예제 3]

화면에 'Input string:'이라는 문자열을 표시한 다음 키보드에서 임의의 문자열을 입력받아 줄을 바꾼후 그 문자열을 출력하는 프로그램을 작성하라.

문자열의 입력은 DOS 기능호출의 0AH를 호출하여 실행한다. 0AH를 호출하려면 데이터를 저장할 위치의 첫번째 번지를 DX레지스터에 설정한다.

또한 데이터를 저장할 첫번째 번지에는 입력받을 최대문자수(0~255)를 설정해 두어야 하며, 그 번지에 0이 들어 있으면 문자열의 입력은 수행되지 않는다. 문자열의 입력은 〈ENTER〉키를 누르면 종료되는데 만일 문자열의 길이가 최대 허용문자 수를 넘어서면 '삑'하는 음을 출력하며 입력을 중지시켜 버린다.

0AH 기능호출을 수행하면 입력된 문자의 수와 함께 입력받은 문자가 메모리에 기억되는데, 실제의 데이터는 메모리 선두번지 +2부터 저장된다.

이 기능호출을 이용하면 템플리트 기능을 이용할 수 있다. 즉 'ABCDEF'를 입력한 후라면 〈F3〉키를 치면 앞에서 입력한 내용('ABCDEF')이 그대로 나타난다는 뜻이다. 템플리트기능이 생소한 독자들은 DOS 매뉴얼을 다시 한 번 살펴보기 바란다. 입력된 데이터가 저장되는 방식을 **그림 4**에 보였다.

입력된 문자열의 출력은 선두번지 +2를 DX 레지스터에 넣고 DOS 기능호출의 9번 기능으로 출력하는데 9번을 호출하려면 문자열의 끝에 '$'가 있어야 함은 앞에서 설명한 바와 같으며 문자열의 길이를 구한 다음 끝에다 '$'를 추가시키는 방법도 있을 수 있으나 아직 레지스터간의 연산명령은 배우지 않았으므로 이 방법은 일단 접어두기로 하자.

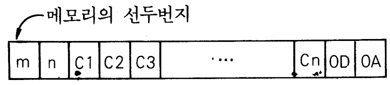
문자열을 입력 받는 부분의 선두 바이트를 제외하고 전체를 '$'로 가득채운 다음 그 위치에다 문자열을 입력받으면 문자열의 다음은 자연히 '$'가 붙을 것이므로 프로그램이 간단해진다.

여기서는 최대 100문자까지 입력받을 수 있도록 하였다. 이때 〈ENTER〉키는 문자수에 포함되지 않을 것이므로 '$'는 최대 103개가 있어야 한다. **리스트 3**을 참고하기 바란다.

어셈블리 프로그램에서는 하나의 프로그램 안에 여러개의 세그먼트를 설정할 수 있는데, 세그먼트의 이름은 사용자 단어 규칙에 맞는 것이라면 아무것이나 좋다. 단지 세그먼트를 설정한다고 해서 세그먼트 레지스터에 값이 할당되는 것이 아니고 그 값이 설정되도록 AX 레지스터 등을 경유해서 해당 세그먼트 레지스터를 초기화 시켜주어야 한다.

리스트 3에서는 MAIN 세그먼트와 DATA 세그먼트를 선언하여 사용하고 있고, 다시 MAIN 세그먼트는 코드 세그먼트로, DATA 세그먼트는 데이

〔그림 4〕 데이터가 저장되는 방식



m : 입력할 최대 문자수
n : 입력된 문자수(리턴코드는 포함되지 않음)
C_1~C_n : 입력된 문자열
0D, 0A : 리턴(Return) 코드

터 세그먼트로 사용하고 있음을 볼 수 있다. 'DATA'는 단순히 사용자 단어일 뿐 이 자체가 '데이터 세그먼트'를 나타내는 것은 아니며 'DATA' 대신에 'KEEP'도 좋고 'ABC'로 해도 좋을 것이다.

리스트 3의 프로그램을 보면 'MAIN ENDS'로 MAIN 세그먼트가 끝나고 있고 그 뒤에 'DATA SEGMENT'로 데이터 세그먼트가 시작되고 있으며 'DATA ENDS'로 데이터 세그먼트가 끝나고 있다.

8088에서 메모리에 있는 데이터를 참조할 경우 잠정적으로 데이터 위치가 데이터 세그먼트내에 있는 것으로 한다는 것을 앞에서 언급했는데 리스트 3에서 번지가 부여되는 것을 보면 0028H 다음에 0000H번지로 되어 있음을 볼 수 있다. 이 0000H번지는 '데이터 세그먼트 레지스터(DS)의 값으로부터 0번째 떨어진 번지'라는 뜻이 된다. 프로그램의 선두를 보면 다음과 같은 내용이 있다.

```
MOV  AX, DATA     ( DATA는 세그먼트의 이름)
MOV  DS, AX
```

〔리스트 3〕 문자열을 입력하는 프로그램

```
0000                              MAIN     SEGMENT
                                           ASSUME  CS:MAIN,DS:DATA
0000  B8 ---- R                            MOV     AX,DATA
0003  8E D8                                MOV     DS,AX
                                           ;
0005  BA 0000 R                            MOV     DX,OFFSET MESG
0008  B4 09                                MOV     AH,9
000A  CD 21                                INT     21H
                                           ;
000C  B4 0A                                MOV     AH,0AH
000E  BA 0010 R                            MOV     DX,OFFSET INPUT
0011  CD 21                                INT     21H
                                           ;
0013  B4 02                                MOV     AH,2
0015  B2 0D                                MOV     DL,0DH
0017  CD 21                                INT     21H
0019  B2 0A                                MOV     DL,0AH
001B  CD 21                                INT     21H
                                           ;
001D  BA 0012 R                            MOV     DX,OFFSET INPUT+2
0020  B4 09                                MOV     AH,9
0022  CD 21                                INT     21H
                                           ;
0024  B4 4C                                MOV     AH,4CH
0026  CD 21                                INT     21H
0028                              MAIN     ENDS
                                           ;
0000                              DATA     SEGMENT
0000  49 6E 70 75 74 20 73        MESG     DB     'Input string : $'
      74 72 69 6E 67 20 3A
      20 24
0010  64                          INPUT    DB     100
0011  0066[                                DB     103  DUP('$')
          24
                       ]

0078                              DATA     ENDS
                                           END
```

이것은 'DATA SEGMENT'라고 선언한 곳의 번지를 DS레지스터에 기억시켜두는 부분이다. 데이터 세그먼트를 코드 세그먼트와 일치시키든 따로 설정하든 데이터를 메모리에 둘 경우에는 공식처럼 기술하는 부분이니 잘 외워두기 바란다.

또 위의 001DH를 보면 'MOV DX, OFFSET INPUT+2'라고 기술된 곳이 있다. 이것은 'INPUT'이라는 레이블이 붙은 번지의 '다음 다음의 위치'라는 것인데 그 앞의 위치를 지정한다면 '+' 대신 '−'를 기술하면 될 것이다.

이번 호의 요약

어셈블리는 베이직만을 사용하는 사람들에게는 무척이나 번거롭고 어려운 언어임에 틀림없다. 베이직같은 고급언어에서는 인간의 언어와 유사한 표현으로 명령을 내리기 때문에 배우는데 별다른 거부감을 갖지 않지만 어셈블리에서는 아직 개념조차 잡히지도 않는 레지스터와 메모리번지를 상대로 하여 프로그램을 작성한다. 또한 대부분의 표현에 있어서 16진수를 사용하기 때문에 16진수를 제대로 모르는 사람들은 상당히 혼란스럽기만 할 것이다.

파스칼과 같은 좋은 언어도 많은데 왜 하필 저급언어를 배우느라고 고생을 해야 하는가 하는 생각을 한두번은 했을 것이다 그러나 어셈블리를 배워두면 그 수고에 대한 댓가는 충분히 보상받고도 남는다. 지금은 다소 고생스럽더라도 참고 견딘다면 머지않아 어셈블리를 자유롭게 구사할 수 있는 날이 올 것이다. 그렇게 되면 IBM-PC는 이제 당신의 손금을 보듯 환해지게 된다.

이번 호에서 설명한 내용들은 공식처럼 중요한 것들이므로 꼭 익혀두기 바란다.

① DB, DW, DD, DQ, DT 등의 의사명령에 의한 데이터 저장
② 데이터가 저장된 위치에 레이블 붙이기
③ 저장된 데이터를 레지스터로 불러오는 방법
④ DS레지스터 설정
⑤ 데이터 세그먼트를 따로 설정하는 방법
⑥ 데이터가 저장된 번지를 레지스터에 기억시키는 방법(OFFSET)
⑦ DUP( ) 의사명령에 의한 문자열의 처리
⑧ DOS 기능호출에 의한 문자열 출력(9번)
⑨ DOS 기능호출에 의한 문자열의 입력(0AH번)
⑩ 램디스크와 배치화일에 의한 어셈블

사운드와 프로그램 분기

이기성/장왕교재연구원 상무

사운드 기능

베이직에서는 음악을 좋아하는 컴퓨터 사용자들을 위해 사운드 기능을 제공하고 있다.

SOUND와 PLAY가 그것인데 비록 명령어는 2개 뿐이지만 여기에 음의 높이와 길이를 변화시킴으로써 다양한 소리를 얻을 수 있다.

(1) SOUND 명령

먼저 사운드 기능 중에서 SOUND에 대해 알아보자. 이 명령어를 이용할 때는 SOUND뒤에 진동 주파수를 숫자(Hertz)로 표시하고 다음에 음의 지속기간을 지정해 준다. 여기서 지속시간이라 하지 않고 지속기간이라한 까닭은 컴퓨터에 내장된 클럭이 보통의 시계처럼 1초에 한번 재깍거리는 것이 아니라 1초에 18.2번을 재깍대기 때문이다. 이 재깍거리는 것을 숫자로 표현하여 SOUND에 지정해 주는데 매뉴얼(IBM Reference Handbook BASIC) 에는 0.0015부터 65535까지 가능하다고 하였지만 테스트한 결과는 0.23부터 65535까지가 가능하였고 0.2이하로 하면 아주 짧은 기간을 소리내는 것이 아니고 아예 무한대의 기간을 소리내고 있었다.

다음 예들을 통해 SOUND명령을 익혀보기로 하자.

```
SOUND 500,10   '낮은음으로 짧게
SOUND 1000,10  '중간음으로 짧게
SOUND 1000,30  '중간음으로 길게
SOUND 2000,30  '높은음으로 길게
```

리스트 1은 SOUND명령어를 이용하여 사이렌 소리, 전화 통화중 소리, 뻐꾸기 소리, 폭탄 떨어지는 소리를 흉내내어 본 것인데 독자 여러분이 음의 높이와 지연기간을 변동시켜 더 실감나게 고쳐서 사용하기 바란다. 참고로 'SOUND 32767, 지연기간'은 소리를 내지 않게 할 때 사용할 수 있다. 그리고 170번 문장과 310번 문장에서 보이는 FOR-NEXT문은 단지 시간을 지연시켜 주는 역할을 하는데 이를 지연루프(Delay Loop)라고 한다.

(2) PLAY 명령

다음은 PLAY명령어에 대해서 알아보자. 이 명령은 음악에 조예가 깊어야만 완전하게 사용할 수 있을 정도로 무궁무진하게 음을 변화시킬 수 있는 장점을 지니고 있는데 지난호의 그래픽 명령에서 배운 DRAW처럼 음의 매크로 스트링을 포함할 수 있는 점이 바로 그것이다.

기본적인 요소들을 보면 음계를 나타내는 기호로서 '도 레 미 파 솔 라 시'를 알파벳 'C D E F G A B'로 표현한다. 이는 더이상 설명할 필요가 없으리라 본다. PLAY "C D E"는 '도 레 미'음을 낸다. 그럼 여기서는 어떤 옥타브의 음을 표현하고 있을까? GWBASIC은 0에서 6까지 7 옥타브를

〔리스트 1〕 SOUND를 이용한 소리

```
100 '사이렌 소리
110 FOR I = 440 TO 1000 STEP 5
120 SOUND I,.5
130 NEXT I
140 '전화 통화중 소리
150 FOR X = 1 TO 5
160 SOUND 440,10
170   FOR K = 1 TO 1200
180   NEXT K
190 NEXT X
200 '빨랐다 느렸다하는 사이렌 소리
210 FOR X = 1 TO 4
220   FOR L = 700 TO -700 STEP -10
230   SOUND 850-ABS(L),.3
240   L = L-2/700
250   NEXT L
260 NEXT X
270 '뻐꾸기 소리 3번
280 FOR X = 1 TO 3
290 SOUND 900,3
300 SOUND 700,4
310   FOR K = 1 TO 1600
320   NEXT K
330 NEXT X
340 '개구리 소리 10번
350 FOR X = 1 TO 10
360 SOUND 700,4
370 SOUND 100,4
380   FOR K = 1 TO 1200
390   NEXT K
400 NEXT X
410 FOR PA = 1 TO 1000:NEXT PA
420 '폭탄 떨어지는 소리
430 FOR K = 3000 TO 600 STEP -5
440 SOUND K,K/5000
450 NEXT K
460 SOUND 3000,2
```

표현할 수 있으며 디폴트로는 옥타브 4를 나타낸다. 그러므로 앞의 예에서도 옥타브 4의 음인 도레미를 출력한다. 다음 예들을 통해 디폴트인 옥타브 4에서의 각 음을 들어 보자.

```
PLAY "C D E"
PLAY "G B D"
PLAY "B A G F E D C"
```

특정 옥타브를 지정할 경우는 알파베트 'O'를 음계 앞에 삽입한다. 예를 들어 O1은 옥타브 1을, O2는 옥타브 2를 의미한다. 다음 보기들을 통하여 각 옥타브에서 출력되는 소리의 차이를 살펴 보기 바란다.

```
PLAY "O0 C D E F G A B"
PLAY "O1 C D E F G A B"
PLAY "O2 C D E F G A B"
PLAY "O3 C D E F G A B"
  :    :       :
```

음계외에 PLAY명령을 조절하는 명령으로서 반음 높게 할 때(Sharp)는 '#' 또는 '+'를 추가하고 반음 낮게 할 경우에는 '-'를 추가한다.

```
PLAY "O2 EGBGEGBGEG-BG-EG-BG-FA# O3DO2A"
```

옥타브 다음으로 중요한 요소가 음의 길이 문제이다. SOUND에서는 진동주파수를 입력했으나 여기서는 알파벳 'L'을 이용하여 해당 수치를 입력해 주면 되는데 입력 형식은 1/n으로 n은 1~64사이의 정수이어야 한다. 즉 L1은 온음, L2는 반음, L3는 3/4박자음, L4는 1/4박자음,… L8은 1/8박자음,… L64는 1/64박자음을 나타낸다. 만일 특정 음의 길이만을 변화시키고자 할 때는 길이를 음뒤에 바로 붙일 수 있다. 이런 경우 'L8A'와 'A8'은 같은 것이다. 또 한가지 재미있는 점은 음이나 연장 기호 뒤에 마침표(.)가 오면 길이에 3/2가 곱해져서 그 만큼 스케일(Scale)된다는 것이다.

```
PLAY "L8 BAGFEDC <BAGFEDC>"
PLAY "L8 T255 BAGFEDC <BAGFEDC>"
PLAY "L64 BAGFEDC <BAGFEDC>"
PLAY "L64 T255 BAGFEDC <BAGFEDC>"
```

음악 연주가 너무 단조롭다고 느낄 때는 PLAY명령에다 'MB'명령을 추가하면 배경(Background) 음악을 연주하도록 한다. 이 명령어 단독으로는 실행이 되지 않은 채 있다가 다음번 PLAY명령어가 나타나면 실행이 된다. 리스트 2에서 2010번행에서 'mb'를 입력하면 2030행부터 작동한다. 음악을 전공하지 않은 사람도 2200행에서 'STOP' 으로 'mb'의 작동이 일단 정지된 상태에서 'RUN 3000'으로 다시 실행시키면 3010행의 연주와 'mb'가 5000행에서 들어간 다음의 5010행의 연주가 다른 것을 알 수 있다.

다음은 X변수를 이용해서 서브스트링을 연주하는 방법을 알아 보자. 이것은 음의 기호들이 모인 스트링을 변수명에 기억시키고 PLAY명령에서 X부명령으로 실행시키는 것이다. 이런 방법을 이용하면 동일한 음들의 모임을 하나의 변수명에 저장하므로 불필요하게 반복되는 프로그램 코딩을 쉽게 해주고 타이핑 오류등을 효율적으로 막아줄 수 있다.

```
10 DATA1$ = "O4 C E G"
20 PLAY "XDATA1$;"
```

이 외에도 PLAY문을 제어해 주는 명령들이 많이 있으며 필요시 이들을 응용하면 되는 것이다(표 1 참조).

리스트 3는 베토벤의 월광곡 앞부분을 코딩한 것으로 표 1을 참조하면서 각 기호들을 이해하기 바란다. 한 예로 110행의 첫부분을 보면 옥타브를 1로 지정하고(O1) 16분쉼표(P16) 만큼 쉬고 G음을 반올림(G+)하여 첫 음을 연주하고 있다.

〔리스트 2〕 배경음악 연주

```
100 PLAY "C D E F G > A B C"
102 SOUND 1000,20
110 PLAY "O3 C D O4 E F G C O3 D E"
112 SOUND 1000,20
120 PLAY "O3 C D O4 E F G C O3 D E"
122 SOUND 1500,20
130 PLAY "O3CDO4EFGCO3DE"
132 SOUND 2000,20
900 PRINT:PRINT "베토벤의 운명중 처음만 ":PRINT
1000 PLAY "P1 O4 E12 E 12 C1 P4 D12 D12 D12 O3 B1"
1100 PLAY "P1 O4 E12 E 12 C1 P4 D12 D12 D12 O3 B1"
2000 FOR X = 1 TO 10
2010 PLAY "MBMN L16 T120"
2030 PLAY "O2EGBGEGBGEG-BG-EG-BG-FA#O3DO2A 1A#O3DO2A 2AO3DO2AFAO3DO2A"
2040 PRINT X, 2*X, 3*X, 4*X
2050 NEXT X
2100 PLAY "CEG"
2200 PRINT:PRINT "   RUN 3000을 하고 리턴을 누르세요 ":STOP
3000 PRINT:PRINT "MB문장 없이 연주하면 느려짐."
3010 PLAY "O2EGBGEGBGEG-BG-EG-BG-FA#O3DO2A 1A#O3DO2A 2AO3DO2AFAO3DO2A"
4000 PRINT:PRINT "다시 MB문장을 넣으면 빠른 연주로 됨."
5000 PLAY "MBMN L16 T120"
5010 PLAY "O2EGBGEGBGEG-BG-EG-BG-FA#O3DO2A 1A#O3DO2A 2AO3DO2AFAO3DO2A"
```

명령어	내용
A - G, C+, G-	도 - 시, 반음 올리거나 반음 내리기
O n	0에서 6까지의 옥타브(Octave)번호지정
L n	음의 길이. n=1은 온음표, n=4는 4분음표
MF	주제 연주 (Music Foreground)
MB	배경 음악 (Music Background)
MN	정상으로 (Music note length Normal)
ML	레가토로 (Music note length Legato)
MS	스타카토로 (Music note length Staccato)
N n	0에서 84 사이를 연주 (piano notation No.)
P n	쉬는 기간, n =1은 온쉼표, n=4는 4분쉼표
T n	32에서 255까지의 박자(Tempo)
X 문자열	문자열의 내용을 실행시킴(Execute)
> n, < n	n옥타브만큼 올리거나 내린다.

〔리스트 3〕 베토벤의 월광곡 앞부분

```
10 '
20 CLS
30 PRINT" ************ 월 광 곡 처음 부분  **************"
100 PLAY "MFMNL16t160P1P1"
110 PLAY "O1P16G+O2C+EG+C+EG+O3C+ O2EG+O3C+EO2G+O3C+E"
120 PLAY "O3G+C+EG+O4C+ O3EG+O4C+EO3MLG+O4C+EMSG+8G+8MN"
130 PLAY "O1P16G+O2CD+G+ CD+G+O3CO2D+G+ O3CD+O2G+CD+"
140 PLAY "O3G+CD+G+O4CO3D+ G+O4CD+O3MLG+ O4CD+MSG+8G+8MN"
150 PLAY "O2C+FG+O3C+O2FG+ O3C+FO2G+O3C+ FG+C+FG+"
160 PLAY "O4C+O3FG+O4C+FO3G+ O4C+FG+MLC+ FG+O5MSC+8C+8MN"
170 PLAY "O2P16MLC+F+AO3MSC+ MLC+F+AO4MSC+ MLC+F+AO5MSC+8C+8MN"
180 PLAY "O2P16MLC+EGO3MSC+ MLC+EGO4MSC+MLC+ EGO5MSC+8C+8MN"
```

조건에 의한 프로그램 분기

프로그램을 공부하면서 명령어도 물론 익혀야 하지만 여러 조건에 의한 프로그램 분기에 대해서도 많은 관심을 가져야 한다. 어떤 명령어 하나를 알고 있는 사실과 그 명령어를 이용하여 자기의 논리를 자유자재로 구사한다는 것은 별개의 문제라는 것을 알아두자. 앞으로 프로그램을 학습하면서 자신의 논리를 구사하고 시험해 볼 수 있는 기회들이 많이 있을 것이다.

여기서는 프로그램 작성시 원하는 조건에 의한 분기에 대해서 살펴 보겠는데 'GOTO xxx'문 같은 무조건 분기명령도 있지만 이 명령은 될 수 있는 한 쓰지 않는 것이 좋다. 무조건에 의한 분기는 자칫 프로그램 논리를 엉망으로 만들수 있으며 또한 프로그램 작성시 좋지 않은 습관만 몸에 배이게 된다.

그러므로 다소 힘들더라도 그 상황에서의 특정 조건에 의한 분기를 만들어 내야 하는데 이런 습관을 지닐 때 나중에는 좋은 프로그램을 만들 수 있으리라 믿는다.

(1) IF…THEN…ELSE

가장 일반적인 형태의 조건 분기명령으로 'IF..THEN..ELSE'가 있다. IF 다음에 원하는 조건이 들어가고 THEN 다음에는 조건에 합당했을 때 수행할 명령이 들어간다. 그리고 ELSE 다음에는 조건에 맞지 않았을 때 수행할 명령이 주어진다. ELSE 이후는 경우에 따라 생략되기도 하는데 이 경우(조건에 맞지 않을 때)는 다음 문장을 수행하게 된다. 이 문장을 우리말로 표현하면 '만약..이면 THEN 이하를 수행하고..이지 않으면 ELSE이하를 수행하라!'와 같이 된다.

```
10 IF EVENT = TRUE THEN Q=1 ELSE Q=0
```

이 표현은 만약 EVENT가 'TRUE'이면 Q의 값으로 1을 정하고 그렇지 않으면 Q의 값으로 0을 설정한다는 말이다.

물론 이 경우 프로그램 내에서 Q의 값이 1이냐 0이냐에 따라 서로 다른 처리를 하게 된다. 다음 예를 보자.

보기 1의 프로그램은 두개의 수를 입력받아 크기를 비교한 후 그 결과를 화면에 디스플레이하는 것이다. 참고로 여기서 세미콜론(;)과 콜론(:)의 용법 및 PRINT문에서 변수명과 실제 표시할 데이터의 쓰임새를 알아 두자.

즉 프로그램 실행시 유동적인 메시지를 화면에 나타낼 때는 고정적인 메시지와 유동적인 메시지를 위와 같이 조합하여 표현하면 편리하다. 컴퓨터는 명령 하나를 처리하고 다음명령을 처리할 때는 잠정적으로 화면의 다음 줄을 기준으로 하기 때문에 이를 연결시킬 때 세미콜론을 이용한다. 즉 ';'으로 연결된 두 명령어는 화면의 같은 줄에 처리결과를 표시하는 것으로 모든 경우에 가능한 것은 아니며 주로 PRINT문에서 쓰인다. 이와는 달리 ':'는 두개의 명령을 분리시키는 작용을 한다. 주로 프로그램 코딩에서 별개의 줄로 된것을 하나의 줄로 표시할 때 쓰이고 있다.

그러므로

```
10 CLS
20 LOCATE 3,10
30 PRINT "BASIC 학습"
```

와

```
10 CLS:LOCATE 3,10:PRINT "BASIC 학습"
```

은 같은 효과를 낸다.

보기 1의 예를 보면 한가지 미흡한 점이 있다. 입력한 두개의 수가 같은 경우를 고려하지 않았다는 점이다. 보기1에서는 같은 수라도 첫번째 비교에서 앞의 수가 뒤의 수보다 크다고 볼 수 없으므로 무조건 뒤의 수가 크다고 출력해 버린다. 이런 경우 IF..THEN.. ELSE논리를 중복해서 표현하면 되는데 한가지 단점이 있다면 베이직에서 이런 중복논리를 여러 줄에 표현할 수 없다는 점이다. 다음과 같이 PRINT문을 분리하여 완전히 비교가 되도록 프로그램을 고쳐 보자(보기 2 참조).

이렇게 되면 문제는 해결되었으나 프로그램이 매우 복잡해져 버렸다. 앞에서도 설명 했지만 ELSE이하를 생략할 수 있고 이런 경우는 (조건에 부합이 안될 경우) 그 다음 문장이 처리되기 때문에 보기 3과 같이 코딩을 분리하면 프로그램을 이해하기가 편리해진다.

〈보기 1〉

```
10 INPUT "첫번째 숫자를 입력하세요:",FIRST%
20 INPUT "두번째 숫자를 입력하세요:",SECOND%
30 IF FIRST% > SECOND% THEN PRINT FIRST%;"가 ";SECOND%;"보다 큽니다.":END
40 PRINT SECOND%;"가 ";FIRST%;"보다 큽니다."
```

〈보기 2〉

```
10 INPUT "첫번째 숫자를 입력하세요:",FIRST%
20 INPUT "두번째 숫자를 입력하세요:",SECOND%
30 IF FIRST%<>SECOND% THEN IF FIRST%>SECOND% THEN 80 ELSE 90 ELSE 100
80 PRINT FIRST%;"가 ";SECOND%;"보다 큽니다.":GOTO 110
90 PRINT SECOND%;"가 ";FIRST%;"보다 큽니다.":GOTO 110
100 PRINT FIRST%;"와 ";SECOND%;"는 같은 수입니다.":GOTO 110
110 END
```

〈보기 3〉

```
30 IF FIRST% = SECOND% THEN PRINT FIRST%;"와 ";SECOND%;"는 같은 수입니다.":GOTO 110
40 IF FIRST% > SECOND% THEN PRINT FIRST%;"가 ";SECOND%;"보다 큽니다.":GOTO 110
50 PRINT SECOND%;"가 ";FIRST%;"보다 큽니다."
110 END
```

〈보기 4〉

```
50 LOCATE 20,10: PRINT "아무 키나 누르면 계속됩니다..."
60 ANS$ = INKEY$
70 IF ANS$ = "" THEN GOTO 70
80 REM 프로그램 계속 실행
```

(2) INKEYS

프로그램 수행 도중 키보드를 통해 사용자로부터 아무 키든 입력받는 명령이 있다. INKEY$가 그것인데 더 정확히 말하면 컴퓨터가 프로그램을 수행하다가 이 명령을 만나면 키보드 버퍼로부터 한문자를 읽어낸다는 것이다. 키보드 버퍼는 키보드로부터 입력되어지는 문자를 잠시 기억해 두는 장소라고만 알아두자. INKEY$는 필자도 즐겨 쓰는 명령으로 재미있는 분기를 만들어 준다.

프로그램 수행 도중 '아무 키나 누르면 계속됩니다…라는 메시지와 함께 잠시 실행을 중지 시켰다가 키를 누르면 프로그램을 계속 수행한다(보기 4 참조).

이같은 경우도 위에서 설명한 IF..-THEN.. ELSE문에서 ELSE이하가 생략된 경우로써 60번 행에서 ANS$ 변수에 한 문자를 입력시키고 70번 문장에서 이를 체크한다. 즉 ANS$에 아무런 키가 입력되지 않으면("":공백 스트링) 70번행을 계속 수행한다. 다시 말하면 ANS$에 입력되는 문자를 계속 체크하게 되는데 이 시간은 프로그램이 잠시 정지한 것과 같은 효과를 얻는 것이다. 이 때 아무 키든지 누르면 ANS$가 공백 스트링이 아니므로 프로그램 제어는 다음 문장 즉 80번으로 옮기게 되어 프로그램은 계속 실행되는 것이다.

(3) INPUT

INKEY$와 비슷한 명령으로 INPUT문이 있다. 보통 이 경우는 키보드로부터 아무키를 원하는 것이 아니라 특정 문자를 원하는 경우에 쓰인다. 쉬운

예로 게임프로그램을 하다가 게임이 끝났을 때 화면에 나타나는 메시지가 그것이다. 보통 다음과 유사한 문장일 것이다.

```
DO YOU WANT PLAY AGAIN ? (Y/N):_
```

즉 게임을 다시 하고 싶으면 Y(Yes: 예)를 누르고 그냥 끝내고 싶으면 N(No:아니오)을 입력한다. 마찬가지 효과를 다른 프로그램에서도 얻을 수 있는 것이다(보기 5 참조).

70번행에서 'Y'의 대문자와 소문자를 논리합(OR)으로 연결시키고 있는데 이는 키보드를 통해 대문자 상태든 소문자 상태든 상관 없이 받아들이기 위한 것이다. 참고로 이와 같이 프로그램에서는 사소한 경우까지 섬세하게 고려해 주어야 한다는 점을 말해 두고 싶다.

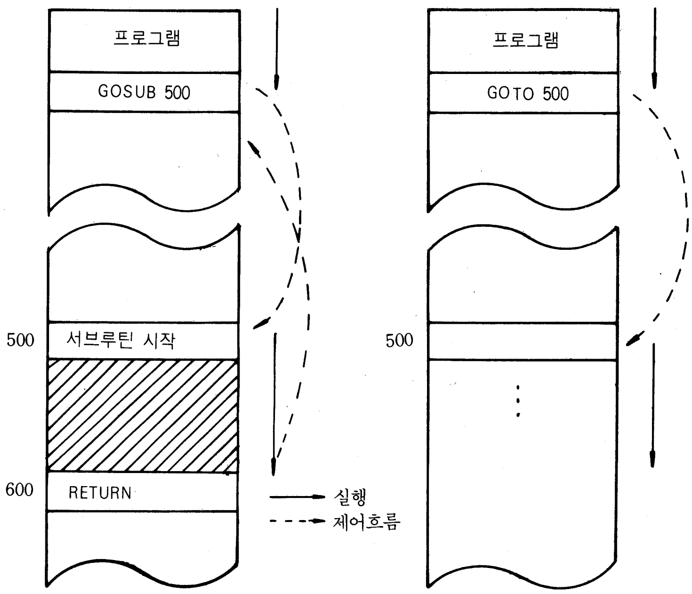
위의 예에서 PRINT문을 없애고 INPUT 문 단독으로 화면 메시지와 함께 원하는 문자를 입력시킬 수도 있는데 60번행을 다음과 같이 변형시키면 된다.

```
60 INPUT "데이터 입력을 다시 하시겠습니까 ?
       (Y/N) :",ANS$
```

〈보기 5〉

```
50 LOCATE 20,10
60 PRINT "데이터 입력을 다시 하시겠습니까 (Y/N) :";:INPUT ANS$
70 IF ANS$ = "Y" OR ANS$ = "y" THEN GOTO xx
80 REM End of the Program
90 PRINT "감사합니다."
```

〔그림 1〕 GOSUB과 GOTO 비교



(a) GOSUB
(지정된 루틴을 실행하고 다시 돌아오는 논리)

(b) GOTO
(지정된 문장번호로 점프하고 계속 실행)

(4) WHILE…WEND

이번에는 조금 다른 형태로써 'WHILE…WEND'문에 대해서 알아 보자. 이 명령은 주어진 조건이 참인 동안만 루프속에 있는 명령을 수행하게 할 때 필요한 것으로 WHILE 다음에 지정한 식을 체크하여 참이면 WEND를 만날 때까지 주어진 명령들을 수행하고 다시 원래 위치로 돌아와 다시 식을 체크한다. 이렇게 반복하다가 식이 거짓으로 판별되면 WEND 다음의 문장으로 분기해 버린다.

```
20 COUNT = 1: SUM = 0
30 WHILE COUNT < 101
40 SUM = SUM + COUNT : COUNT = COUNT + 1
60 WEND
70 PRINT "합계 = ";SUM
```

(5) ON…GOSUB

대부분의 프로그램을 보면 각 상황에서 선택메뉴를 나타내어 사용자가 원하는 기능번호를 입력한 후 해당작업을 할 수 있게 하는 경우가 대부분이다. 이 때의 프로그램 분기는 서브루틴, 또는 규모가 큰 프로그램인 경우 여러화일 사이에서 행해진다. (이 경우는 메인프로그램을 작성해서 이들 각 루틴 또는 각 화일사이의 분기를 제어하도록 하면 좋다.)

'ON..GOSUB..' 또는 'ON..GOTO..' 문을 이용하면 한번에 수개의 분기 명령을 선택을 할 수 있게 해준다. ON 다음에 정수형 변수가 오는데 그 숫자에 따라서 GOSUB 또는 GOTO 뒤에 나열된 분기 장소로 제어가 옮겨진다.

GOSUB과 GOTO의 차이점은 그림 1을 참조하기 바란다. 다음 예의 경우 INPUT문으로 IN_P$변수에 해당 기능 번호가 입력되긴 하나 숫자 상태가 아닌 문자 상태로 입력된다는 것을 알아두자. 그래서 문자 상태로 기억된 수

〔화면 1〕 메뉴 선택방법(1)

```
********* M E N U *********
*                         *
*    1. CREATE THE FILE   *
*    2. UPDATE THE FILE   *
*    3. OUTPUT THE FILE   *
*    4. EXIT              *
***************************
```

를 수치형태로 변형시켜 주는 명령어가 필요한데 이것이 바로 VAL 함수이다. VAL은 괄호 속의 스트링을 수치값으로 만들어 준다. 이렇게 해서 얻은 수치를 ON 다음에 집어 넣어 그 값에 따라 GOSUB 뒤의 해당 문번호로 분기하는 원리이다. 참고로 화면 1의 내용을 리스트 4에 나타내었다.

```
100 LOCATE xx,yy
110 INPUT$"원하는 기능번호를 선택하세요:";IN_P$
120 NUM% = VAL(IN_P$)
130 ON NUM% GOSUB 1000,1500,580,2000
:
```

즉 위에서 NUM%정수형 변수의 값이 1이면 1000번행으로 분기하고, NUM%의 값이 2이면 1500번행, 3이면 580번행, 끝으로 4이면 2000번행으로 분기하게 된다.

(6) CSRLIN을 응용한 매뉴선택

이번에는 메뉴선택에 따라 분기시키는 독특한 방법에 대해 알아보자. 즉 지금까지는 화면 1처럼 메뉴에 선택번호를 할당하여 원하는 번호를 입력받아 해당루틴으로 분기시켰으나 GWBASIC 의 CSRLIN이란 함수를 적절히 응용하면 키보드의 상하 화살표를 이용하여 메뉴를 선택할 수 있다. 보통 원하는 메뉴앞에서 커서가 ■의 형태로 깜박거리게 하고 상하화살표를 이용하여 커서를 이동시킨다. 원하는 위치에 커서가 오면 〈ENTER〉키를 눌러 선택한다(화면 2 참조).

〔화면 2〕 메뉴 선택방법(2)

```
*********  M E N U  *********
*                           *
*      █CREATE THE FILE     *
*      _UPDATE THE FILE     *
*      _OUTPUT THE FILE     *
*      _EXIT                *
*****************************
```

〔리스트 4〕 SELECT.BAS

```
10 '
20 CLS:SCREEN 0
30 LOCATE 3,20:PRINT "********* M E N U *********"
40 LOCATE 4,20:PRINT "*                          *"
50 LOCATE 5,20:PRINT "*      1. CREATE THE FILE  *"
60 LOCATE 6,20:PRINT "*      2. UPDATE THE FILE  *"
70 LOCATE 7,20:PRINT "*      3. OUTPUT THE FILE  *"
80 LOCATE 8,20:PRINT "*      4. EXIT             *"
90 LOCATE 9,20:PRINT "***************************"
100 LOCATE 20,10
110 INPUT "원하는 기능번호를 선택하세요:",A
130 ON A GOSUB 1000,1500,580,2000
140 GOTO 20
580 '
590 RETURN
1000 '
1010 RETURN
1500 '
1510 RETURN
2000 '
2010 BEEP:END
```

〔리스트 5〕 MENU.BAS

```
10 'Sampple Program
20 '---- Using of the CSRLIN command ----------------------------------
40 GOSUB 200
50 X%=5: Y%=26: LOCATE X%,Y%,1,0,8
60 FIRST%=5: LAST%=8
70 ANS$=INKEY$: IF ANS$="" THEN 70
80 IF ANS$=CHR$(0)+CHR$(80) THEN IF X%<LAST% THEN X%=X%+1 ELSE X%=FIRST%
90 IF ANS$=CHR$(0)+CHR$(72) THEN IF X%>FIRST% THEN X%=X%-1 ELSE X%=LAST%
100 LOCATE X%,Y%: IF ANS$<>CHR$(13) THEN 70
110 FLAG1%=CSRLIN: FLAG1%=FLAG1%-4
120 ON FLAG1% GOSUB 1000,1500,580,2000
130 IF FLAG1%<>1 THEN GOTO 40 ELSE GOTO 50
200 'Create the Menu Display
210 CLS:SCREEN 0
220 LOCATE 3,20:PRINT "*********  M E N U  *********"
230 LOCATE 4,20:PRINT "*                           *"
240 LOCATE 5,20:PRINT "*      _ CREATE THE FILE    *"
250 LOCATE 6,20:PRINT "*      _ UPDATE THE FILE    *"
260 LOCATE 7,20:PRINT "*      _ OUTPUT THE FILE    *"
270 LOCATE 8,20:PRINT "*      _ EXIT               *"
280 LOCATE 9,20:PRINT "*****************************"
290 LOCATE 20,20: PRINT "Select the correct Issue."
300 RETURN
580 '------- OUTPUT THE FILE
590 REM The method of output the file
600 RETURN
1000 '------ CREATE THE FILE
1010 REM This routine contains creating the file
1020 RETURN
1500 '------ UPDATE THE FILE
1510 REM This routine contains updating the file
1520 RETURN
2000 '------ EXIT
2010 REM Exit the file
2020 BEEP
2030 END
```

GSRLIN은 현재 커서가 위치한 세로라인 위치를 구해주는 함수이다. (POS 함수를 이용하면 커서의 칼럼위치를 구해 준다.) 먼저, 커서모양을 ■로 하기 위해서 LOCATE명령의 파라미터 옵션을 선택해야 한다. LOCATE 명령의 완전한 형식은

LOCATE [행],[열],[커서지정],[시작],[끝]

인데 보통은 행과 열만을 사용하여 화면상의 위치만 지정했으나 뒤의 파라미터를 첨가하면 프로그램 실행시 커스를 보이게 할 수도 있고 (평상시는 프로그램 실행시 커서가 안보이게 되어 있다.) 커서의 모양을 마음대로 변형시킬 수도 있다(아무 지정이 없을 때 커서 모양은 '—'이다.).

[행] : 화면상의 행번호를 표시
[열] : 화면상의 열번호를 표시
[커서지정] : 프로그램 실행중 보이게 하느냐 안 보이게 하느냐를 결정 (0이면 커서가 꺼지고 1이면 켜짐)
[시작] : 커서모양을 그리는 시작 스캔라인
[끝] : 커서모양을 그리는 끝 스캔라인

여기서는 완전한 직사각형 커서를 만들기 위해 LOCATE,,,0,8을 사용하기 바란다(리스트5 참조).

리스트 5의 80번과 90번 문장에서 CHR$(80), CHR$(72)가 보이는데 이것은 상하 화살표키를 나타내는 ASCII 코드이고 CHR$(13)은 리턴키를 나타내는 ASCII코드이다. PC

프로그램의 설계와 개발

이용일/한국전자통신연구소

퍼스널컴퓨터의 표준 운영체계로서 지위를 확보한 MS-DOS는, 퍼스널컴퓨터의 이용환경으로서 다른 운영체계와는 다른 의미를 갖고 있다. 현재의 MS-DOS라고 하는 DOS 환경에서는 여러가지 다른 목적으로 이용하는 다방면의 사용자가 있고, 여러종류, 여러형태의 소프트웨어, 하드웨어가 판매되고 있지만, 그 시스템의 형태는 모두 같지가 않다.

MS-DOS가 제공하는 이와같은 환경은, 대부분의 사용자에게 이익을 가져다 주지만, MS-DOS에서 소프트웨어를 만들려는 프로그램 작성자에게는 보통의 프로그램 개발에 얽혀있는 문제에 부가하여 몇개의 특이한 문제를 부담시키게 된다. 즉, MS-DOS의 환경에 적합한 개발환경이란 무엇인지 그리고 하드웨어에 의존하지 않기 위한 프로그램의 범용성 및 이식성, DOS의 버전갱신에 대한 대응책 마련등은, MS-DOS라는 독자적인 환경에서는, 새로이 반드시 고려하여야 하는 것이다.

이번회에서는 우선 프로그램 개발의 전단계로서, MS-DOS에서의 프로그램 개발시 고려해야하는 여러 관점에 대해 검토하기로 한다.

MS-DOS에서의 프로그램 작성

MS-DOS에서 프로그램을 개발할때는, MS-DOS자체의 규격 및 MS-DOS를 둘러싸고 있는 소프트웨어, 하드웨어를 충분히 고려할 필요가 있다. 예를 들면, 처리계 하나의 선택을 생각해 보아도, 현재 판매되고 있는 수많은 처리계중에서, 프로그램의 처리내용 및 규모에 부응하는 언어를 선택할 필요가 있고, 각종의 개발도구들도 준비해야만 한다.

실제 프로그램 개발에 앞서서, 문제로서 생각해야 하는 것은 개발환경의 정비와 함께 프고그램의 규격 및 특성의 결정이다. 프로그램의 목적에 따라, 그 규격 및 특성에 대해 충분히 검토해야 할 필요가 있다. 수명주기가 긴 프로그램이면, 범용성, 이식성, 보수성등이 프로그램 개발상의 중요한 문제가 된다. 또, MS-DOS의 앞으로의 버전갱신에 대한 대응 및 마이크로소프트 윈도우 및 마이크로소프트 네트워크라는 확장된 환경까지도 고려사항에 넣으면 그래픽스 처리 및 화일 공유에 대해서도 프로그램의 규격 및 특성을 신중히 결정하여야 한다.

또, 이들의 범용성 문제와 관련하여 MS-DOS가 제공하는 서비스의 범위내에서 그 기능을 충분히 활용하기 위해서는, 프로그램 자신이 어느정도의 기교를 구사해야 한다. 예를 들면, 필터와 같이 DOS에 의존하는 기능을 거의 이용하지 않는 이식성이 높은 것을 작성하는 경우에는, C언어의 표준 라이브러리만을 이용하여 작성할 수 있지만, MS-DOS와 밀접한 기능을 지원하기 위해서는, C 언어로 프로그램을 작성한다 하여도, 펑션콜의 직접호출 및 어셈블러 루틴과의 링크등, 하위수준의 DOS 및 프로세서에 밀착한 기법을 적극적으로 살려야 할 필요가 있다.

프로그램 개발언어의 선택과 개발도구

프로그램을 개발할 때에는, 컴파일러를 비롯하여, 디버거, 에디터 등의 각종 개발도구를 이용하는 것이 개발효율은 물론 그 프로그램의 이식성 및 보수의 용이성, 이용할 수 있는 시스템의 제한등, 여러가지 요소에 큰 영향을 미친다. 예를 들면, 과학기술용 수치계산 프로그램 등에서, 장래 대형계산기에서 실행시킬 가능성이 있는 것이라면 포트란에 의해 프로그램을 개발하여야 하고, 게임용 소프트웨어이면, 속도가 생명이므로 이식성을 무시하더라도 어셈블리 언어 만으로 모든 프로그램을 작성하도록 해야 한다. 여기서는 C 언어 및 개발도구를 소개하고, 그 기능의 해설과 함께 실제의 사용실례를 보일 것이다.

1. 개발언어의 선택

시스템 프로그램에서는 실행속도와 함께, 그 운영체계의 기능을 완전히 사용하기 위해 어셈블리 언어를 이용하는 것이 일반적이다. 그러나 어셈블리 언어로 개발한 프로그램은 제닉스로의 이행과 같은 이식성 및 보수성(어셈블리 언어는 자유도가 높고, 장황하여 버그가 발생하기 쉽다) 등의 문제점이 있어, 속도등으로 극단적인 제약이 있는 경우 이외에는 개발효율의 면에서

도 간편한, 고급언어를 이용하는 것이 유리하다.

현재 C 언어가 각광을 받고 있는 것도 C 언어가 위에 열거한 조건을 만족하는 고급언어의 하나이기 때문이다. C 언어는 원래 UNIX의 주력언어인 것 때문에 주목을 모았던 언어이지만, UNIX와 같은 운영체계를 기술할 수 있을 정도의 능력을 갖고 있고, C 언어가 개발된 당시의 미니컴퓨터와, 현재의 퍼스널컴퓨터의 성능이 대체로 같아 현재는 프로그램 개발용으로 가장 많이 이용되고 있는 언어의 하나이다. 특히 MS-DOS에서는, MS-DOS가 제닉스와의 호환성을 고려하여 설계되고 있는 면도 있어, C 언어와의 호환성도 좋고 또, 많은 대부분의 소프트웨어 제작회사에서 C 언어 처리계를 제공하고 있는 등, 현재 프로그램 개발 언어로서는 C 언어가 가장 사용하기 편리한 언어이라고 말할 수 있을 것이다.

2. C언어에 의한 소프트웨어 개발

C 언어는 컴파일러형식의 언어이기 때문에 실행속도, 프로그램크기의 면에서는 어셈블리 언어에 비해서는 좋지 않다. 그래도 타른 고급언어와 비교하면 고속이고, 처리계에 부속한 풍부한 라이브러리를 이용하는 것에 따라, 어셈블리 언어를 이용하는 경우에는 지원받지 못하던 여러 지원루틴을 쉽게 실현할 수 있어, 라이브러리를 이용하고 싶지 않은 경우에는 스스로 라이브러리를 구축할 수 있는 특징을 갖고 있다.

특히 소프트웨어 개발도구와 같은 경우에는, 프로그램 내부의 처리에 대하여 입출력에 소요하는 시간이 많아, C 언어와 어셈블리 언어의 처리속도에 그다지 차이가 나지 않기 때문에, 보수성 및 개발의 용이성에서도, 프로그램의 대부분은 C 언어로 기술하는 것이 바람직하다고 할 수 있다. 또, C 언어에서는 어셈블리 언어로 씌어진 루틴과도 쉽게 링크할 수 있으므로 속도 및 기술능력의 문제가 있으면 반드시 필요한 부분만을 어셈블리 언어로 기술하고, 그외의 대부분에 대해서는 C 언어에 의해 기술한다고 하는 방법으로, 보다 사용하기 쉬운 프로그램을 작성할 수 있다.

C 언어는 자주 "고급 어셈블러"라고 불려진다. 이 표현은 C 언어의 특성 및 이용형태를 단적으로 표현한, 그야말로 정확한 표현이라고 할 수 있다. C 언어는 고급언어이고, 목적지향의 기술능력을 가진 한편, 포인터(프로세서의 어드레스)를 직접 표현할 수 있는 등 어셈블리 언어에 필적하는 유연성을 겸비하고 있다. 실제의 사용방법도, 데이터베이스 및 컴파일러 등 종래부터 고급언어가 이용되어온 분야에 부가하여, UNIX라는 운영체계의 대부분을 기술했다든지, MS-DOS의 많은 유틸리티 개발에 이용되는 등 지금까지는 어셈블리 언어가 이용되고 있던 분야에서도 적극적으로 이용되고 있다.

(1) 고급언어로서의 C 언어

고급언어에 의한 프로그램 개발노력은, 대부분의 경우 어셈블리 언어에 의한 개발노력의 수십 퍼센트에 불과하다. 그러나, 컴파일러의 성능 및 하드웨어 능력의 균형등에 문제도 있어, 퍼스널컴퓨터상에서의 고급언어에 의한 프로그램 개발의 비율은, 미니컴퓨터 이상의 시스템에 비하면 그다지 높지 않았었다. 그러나, 퍼스털컴퓨터의 성능은 계속 향상하고 있고, 프로세서능력의 향상(8비트→16비트, 속도의 향상 등), 주기억의 증가(64K→1-16M), 하드디스크의 보급등, 고급언어를 이용하기 쉬운 환경이 서서히 형성되고 있다.

또, 특히 MS-DOS의 경우에는 입출력조작의 대부분을 운영체계가 관리하고 있기 때문에 입출력을 직접 조작할 필요성이 적고, 그와같은 점에서도 어셈블리 언어에 의한 프로그램 개발의 필연성은 감소하고 있다고 말할 수 있을 것이다.

프로그램의 실행속도 및 크기 등의 면에서도 보면 고급언어는 어셈블리 언어에 비하여 열등하지만, 그 대신에 고급언어의 특징으로서 프로그램의 작성이 쉬워, 개발, 보수, 버전갱신 등의 주기를 단축할 수 있다는 장점이 있다. 또 고급언어를 이용하는 것에 의해 얻어지는 또 하나의 이점으로서, 프로그램의 이식성이 높아지는 것을 들 수 있다.

어셈블리 언어에 의해 기술한 프로그램은 거의가 목적으로 하는 시스템에 크게 의존하여 이식성도 거의 없다. 예를 들어, 어떤 프로세서용의 어셈블리 언어로 작성된 프로그램은 다른 프로세서상에서는 이용할 수 없고, 운영체계의 기능을 이용하고 있는 경우에는, 다른 운영체계에 이식할 때도 곤란이 따른다. 또 운영체계를 이용하지 않고 하드웨어를 직접 조작하는 프로그램은 프로세서가 같다고 해도 하드웨어구성이 다른 컴퓨터에서는 이용할 수 없을 것이다. 이것에 대하여 고급언어의 경우에는 이와같은 문제의 대부분을 각 시스템마다 준비된 처리계의 라이브러리 내에서 흡수할 수 있기 때문에, 언어 규격 및 라이브러리 규격이 같은 처리계만 준비되어 있으면, 소스 프로그램 수준에서는 상당한 이식성을 가질 수 있게 된다.

고급언어로서의 C 언어는, 근대적인 제어구조 및 데이터 구조의 개념을 채용하고 있어, 어셈블리 언어 및 베이직 등과 비교하여도 프로그램의 작성능력이 뛰어나고, 특히 퍼스널컴퓨터상의 일반적인 수백-수만 라인의 프로그램 기술성이 우수하여, 프로그램의 유지, 보수가 간편하다. 예를 들어, 데이터구조로서 구조체 형 및 공용체 형이, 제어구조로서는 함수정의 및 재귀호출, 스위치문 등이 지원되고 있어 프로그램을 구조적으로 작성할 수 있다. 또, 이식성에 대해서도, C 언어는 원래 미니컴퓨터 및 퍼스널컴퓨터 등, 비교적

소규모 시스템을 중심으로 보급해 온 언어이지만, 최근에는 미국 IBM사에서 대형컴퓨터용 C 컴파일러를 판매하는 등, C 언어를 이용할 수 있는 환경을 포함하여 C 언어가 관장하는 범위도 급속히 넓어지고 있다.

이와같이 기술성능, 이식성의 양면에 대해, 퍼스널컴퓨터의 소프트웨어 개발언어로서 C 언어를 이용하는 것이 효율적이다.

(2) 고급 어셈블리 언어로서의 C 언어

C 언어는 포트란 및 알골 등의 다른 일반적인 고급언어와 달리 개발 당시부터 어셈블리 언어적인 이용을 고려하여 만든 언어이다. 또, C 언어가 개발된 시스템은 현재의 퍼스널컴퓨터와 대체로 같은 정도의 능력과 규모인 미니컴퓨터 이었기 때문에, 컴파일러의 부담이 크지 않도록 언어규격도 가능한 한 최저의 필요사항만을 갖도록 억제하였다.

이 결과 C 언어는 퍼스널컴퓨터상에서 사용해도 그다지 무리가 없는 처리계가 되었고, 또 언어규격이 간결하여 처리계의 작성도 쉽게 되어, C 언어의 보급에 크게 공헌하였다. 이와같은 C 언어규격의 간결성과 어셈블리 언어적인 특징을 요약하면 다음의 3개를 들 수 있다.

a. 강력한 포인터 기능(프로세서의 어드레스조작)을 갖고 있다.
b. 입출력을 언어규격에 포함하지 않는다.
c. 프로세서에 의존하기 쉬운 부분을 무리하게 통일하고 있지 않다.

C 언어의 포인터는 다른 언어와 비교하면, 대단히 강력하고, 프로세서의 내부기능에 가깝기 때문에, 대부분의 경우 프로세서의 논리 어드레스(기계어수준의 어드레스)가 그대로 포인터의 값으로서 취급되고 있다. 그때문에, 포인터를 사용하여 메모리 중의 임의의 장소를 접근할 수 있어, 종래의 고급언어에서는 기술이 어려웠던 디버거 등을 비교적 쉽게 기술할 수 있게 되었다.

또, 입출력은 언어규격에 있어서 운영체계 및 하드웨어 등에 가장 의존하기 쉬운 부분이지만, C 언어의 경우 입출력에 관하여 일체 언어규격으로 규정하고 있지 않고, 입출력을 모두 라이브러리로서 분리하고 있기 때문에 문법적으로는 입출력이 처리계에 의존하는 것은 없다.

이것에 대하여 C 언어 이외의 일반적인 고급언어에서는 입출력은 언어규격의 일부가 되어 있다.

이것은 프로그램의 이식성을 엄밀히 보증하기 위해서는 필요한 것이지만, 그 반면 처리계가 복잡하게 된다든지, 입출력 구문이 기종에 의존하게 되는 문제점이 있다. C 언어의 경우에는 반대로 규격이 다르게 되어 있다고 해도, 해당 함수만을 다시 기술하는 것으로 이식할 수 있다.

입출력 이외에 언어규격이 운영체계 및 프로세서에 의존하기 쉬운 부분으로서는, 기본적으로 준비되어 있는 데이터 형의 종류 및 그 크기를 들 수 있다. C 언어의 경우, 처리계가 기본적으로 준비해야 할 데이터 형의 종류는 규정되어 있지만, 그들의 형이 실제로 어떠한 크기를 갖는가에 대해서는 처리계에 의존하게 되어 있다.

이것은, 프로그램의 이식성 면에서는 다소 문제가 있는 것이지만, 각 처리계는 목적으로 하는 프로세서에서 가장 자연스런 크기의 데이터를 취급하기 때문에, 비교적 고속으로 콤팩트한 코드를 생성할 수 있다.

그 이외에도 C 언어의 어셈블리 언어적인 부분은 다음과 같은 것이 있다.

a. 논리형을 갖지 않는다.
b. 실행시 형 검사를 하지 않는다.
c. 강제 형변환(캐스트)이 가능하다.
d. 비트조작이 가능하다.

이들은 어떤 의미에서는 프로그램을 폭주시키는 등의 대단히 위험한 조작을 야기할 수 있지만, 잘 사용하면 콤팩트하게 고속인 프로그램을 작성할 수 있고, 하드웨어 및 운영체제의 기능을 직접 이용하여 강제 형변환 및 비트조작도 할 수 있다. 또 프로토타입 선언에 의해 형 검사를 강화할 수 있어, 필요에 따라 엄밀한 기술과 유연한 기술을 나누어 사용할 수 있다.

MS-DOS와 같은 퍼스널컴퓨터의 운영체계상의 소프트웨어에서는, 종종 하드웨어에 의존한 저수준 처리를 필요로 하는 경우가 있기 때문에, 이와같은 C 언어의 강력하고 세세한 기능의 양호함이 퍼스널컴퓨터에서의 소프트웨어 개발시 큰 장점이라 생각한다.

C언어 개발환경 및 지원 도구

C 언어는 UNIX와 함께 자라온 언어이기 때문에 UNIX상에서 이용하는 것이 가장 효율적이다. 앞으로 UNIX는 일반의 퍼스널컴퓨터에서도 손쉽게 이용할 수 있도록 될 것이고, 이미 MS-DOS에서도 UNIX의 지원도구들이 계속 이식되고 있어 C 언어 프로그램 지원도구로서 이용할 수 있다.

C 언어에 의한 프로그램 개발의 순서는 보통의 소프트웨어 개발과 마찬가지로 우선 프로그램의 설계부터 시작하여 에디터에 의한 원시화일의 입력, 그리고 컴파일, 링크, 디버그 라고 하는 몇개의 정형적인 순서를 밟는다. 그리고 C 컴파일러의 능력을 최대한 활용하고, 이들 순서를 효율적으로 실행하기 위해서는 우선, 개발환경인 디렉토리 구성 및 이용하는 지원도구들의 장비가 필요하다. 그리고 컴파일러 및 링커, 디버거의 사용법을 잘 이해하여 사용에 능숙할 필요가 있다.

여기서는, 본 연재에서 주로 사용하는 컴파일러, 링커, 디버거, 그리고 라이브러리 관리를 위한 라이브러리 매니저 등에 대해 간단히 설명하고 실제 사용 예를 보이기로 한다. 구체적인 옵

션 및 방법 그리고 상세한 설명 등은 관련서적을 참고하기로 한다.

1. 에디터

먼저 원시화일을 작성하기 위해서는 에디터가 필요하다. MS-DOS에는 EDLIN이란 라인 에디터가 포함되어 있어, 배치화일 등의 비교적 짧은 프로그램을 작성하는 경우에는 손쉽게 사용할 수 있지만, 원시 프로그램 길이가 20-30 라인을 넘으면 사용하기가 불편해진다. 프로그램을 개발, 디버그할때의 에디터의 역할은 간단히 원시라인을 입력하면 그만이라는 의미뿐이 아니기 때문에, 역시 많은 기능을 가진 스크린 에디터가 필요하다. 워드스타, 워드마스터, 보석글 등의 수많은 스크린 에디터가 있으며, 프로그램 개발효율을 향상시키기 위해서는 이들 에디터의 사용방법을 숙지하여야 한다.

프로그램개발 순서를 밟기위해서 화면 1과 같은 프로그램을 작성한다. 화일명은 pc1.c로 하고 프로그램 내용은 "Hello!"라는 문자열을 출력하는 간단한 프로그램이다.

2. 컴파일러

마이크로소프트 C 처리계에는, MSC와 CL이라는 2개의 컴파일러 드라이버(컴파일 제어 프로그램)가 준비되어 있고, 래티스 처리계에는 LC라는 것이 준비되어 있다. 여기서는 마이크로소프트 C 처리계에 대해 중점적으로 설명한다. 컴파일러 드라이버란 실제로 컴파일 처리를 실행하는 프로그램(마이크로소프트 C에는 C1.EXE, C2.EXE, C3.EXE가 있고, 래티스 C에는 LC1.EXE, LC2.EXE 등이 있다.)을 차례로 호출하여 처리를 한다든지, 사용자가 지정한 화일 및 옵션을 해석하여, 그들의 프로그램에 넘긴다든지 하는 처리를 하는 것이다. 이들 2개의 컴파일러 드라이버에 의한 컴파일방법을 소개하고, C 언어에 의한 프로그램 개발 순서의 실제를 밟아본다.

〔화면 1〕 프로그램의 작성예

```
C>edlin pc1.c⏎
New file ---- 기존의 화일이 없어 신규로
*i⏎             작성됨을 의미한다.
        1:* include <stdio.h>⏎
        2:*⏎
        3:*main()⏎
        4:*{⏎
        5:*printf("Hello !\n");⏎
        6:*}⏎
        7:*^Z ----- 입력을 종료하기 위해
*e⏎                 CTRL-Z를 친다.
 |
 └--- 화일을 세이브한다.
C>
```

(1) MSC 명령어의 방법

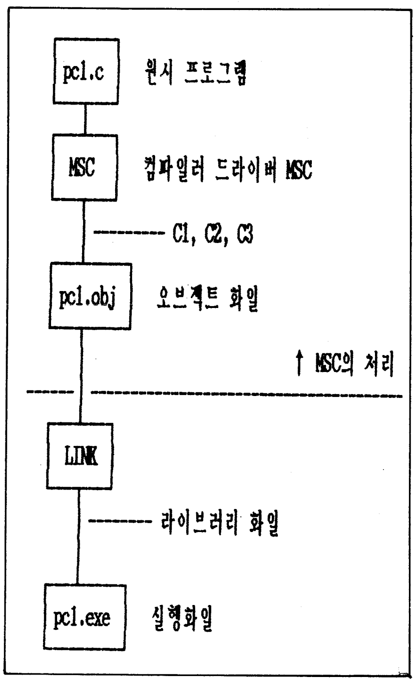
MSC 명령어를 사용한 경우의 기본적인 개발순서는 그림 1과 같다. 이 MSC 명령어는 초보자 지향의 컴파일러 드라이버로서 다음과 같은 몇개의 특징이 있다.

a. 링크를 하지 않고, 오브젝트 화일을 출력한 후 실행을 정지한다.

b. 명령어 한라인으로 모든 지정을 할 수도 있고, 프롬프트를 하나씩 표시하면서 응답해 가는 대화방식으로도 기동할 수 있어 초보자도 간단히 사용할 수 있다.

c. 하나의 원시화일만을 받아 들여 하나의 오브젝트 화일을 출력한다.

〔그림 1〕 MSC 명령어에 의한 개발



MSC 명령어를 기동시켜 보자. 프롬프트에 의한 방식으로 기동시키면 다음과 같은 사용자의 응답을 요구한다.

```
Source filename [.C] :
              ------- 원시화일명의 입력
Object filename [xxx.OBJ] :
              --- 오브젝트 화일명의 입력
Source listing [NUL.LST] :
              ---- 원시 리스팅 화일명의 입력
Object listing [NUL.COD] :
              --- 오브젝트 리스팅 화일명의 입력
```

여기서 화일명의 확장자는 생략할 수 있다. 또 원시화일명의 입력은 반드시 필요하지만, 그후는, 단순히 리턴키 또는 ";"키를 눌러 디폴트의 응답을 사용할 수 있고, 디폴트의 응답을 사용할 때, 오브젝트 화일명의 베이스명에서는 원시화일명의 베이스명과 같은 것이 사용된다.

명령어라인 방식은 다음과 같다.

```
MSC <원시화일명>,<오브젝트화일명>,
<리스트화일명1>,<리스트화일명2>[;]
```

여기서, 마직막의 ";"가 지정되어 있지 않는 경우에는 생략한 화일명을 위해 프롬프트가 표시되어 입력을 요구한다. 이때는 반드시 프롬프트방식과 동일한 응답방식을 사용한다.

화면2에 MSC 명령어의 사용예를 보여준다.

(2) CL 명령어의 사용방법

CL 명령어는 컴파일러 링크를 별도로 해야 하는 MSC 명령어와 달리, 보

다 일반적으로 모든 처리를 실행할 수 있는 명령어이다. 여기서 CL이란 "Compile & Link"의 약어이지만, 이 2개의 처리는 항상 따라다니므로, 분할 컴파일을 하는 경우에는 컴파일만을, 또는 링크만을 할 수도 있지만 CL 명령어는 이들 모든 경우에 대처할 수 있도록 만들어져·있기 때문에 기본적으로 모든 처리가 가능하다. CL 명령어의 특징은 다음과 같다.

a. 에러가 발생하지 않으면 링크까지 하여 실해가능 화일을 출력한다.
b. 프롬프트에 의한 대화형식으로 명령어를 입력할 수는 없다.
c. 명령어라인에서 복수의 원시화일을 지정할 수 있다.
d. 명령어라인에서 복수의 오브젝트화일을 지정할 수 있다.
e. 컴파일 옵션과 함께 링크 옵션을 지정할 수 있다.

당연한 얘기이지만, 개발노력 및 순서는 MSC 명령어보다 간단하게 된다. 그림 2는 CL 명령어에 의한 기본적인 개발순서의 개념도이다.

CL명령어의 일반적인 서식은 다음과 같다.

```
CL 〈옵션1〉 〈화일명〉 -link 〈옵션2〉
```

여기서, 〈옵션1〉은 컴파일 옵션이고, 〈옵션2〉는 링크 옵션이다. 〈화면3〉에 CL 명령어의 사용예를 보여준다.

3. 링커(LINK)

링커는 별도로 어셈블 혹은 컴파일된 오브젝트모듈을 결합하여 외부참조의 해결을 시도하고, MS-DOS상에서 실행가능한 ".EXE"형식의 화일을 출력한다. 링커를 기동할때는 프롬프트 응답형식과 명령어라인 방식 그리고 응답 화일이라 불리는 명령어 화일에 의해 필요한 지정을 하는 방식 그리

〔화면 2〕 MSC 명령어의 기동

```
C>msc↵ —— 프롬프트 방식으로 해본다.
Microsoft (R) C Compiler Version 4.00
Copyright (C) Microsoft Corp 1984, 1985, 1986.  All rights reserved.

Source filename[.C]: pc1↵  —— 확장자없이 화일명만 써도 된다.
Object filename[pc1.OBJ]: ↵
Source listing [NUL.LST]: ↵ ┐
Object listing [NUL.COD]: ↵ ┘ 리턴키만을 눌렀으므로 화일이
                              만들어지지 않는다.
C>dir pc1.*↵
 Volume in drive C has no label
 Directory of  C:\LANG\MSC4

PC1      C        58  1-25-89   9:25a
PC1      OBJ     313  1-25-89  10:13a
        2 File(s)  16224256 bytes free

C>msc pc1,pc1,pc1,pc1;↵ —— 명령어라인 방식으로 한다.
Microsoft (R) C Compiler Version 4.00
Copyright (C) Microsoft Corp 1984, 1985, 1986.  All rights reserved.

C>dir pc1.*↵
 Volume in drive C has no label
 Directory of  C:\LANG\MSC4

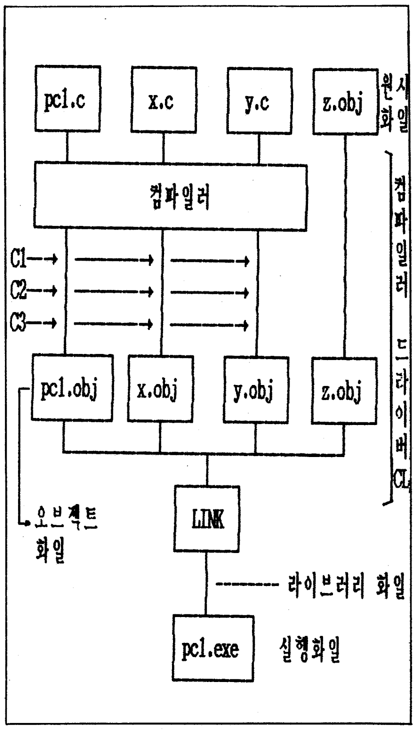
PC1      C        58  1-25-89   9:25a  ┐
PC1      OBJ     313  1-25-89  10:20a  │ 명령어라인의 각각에
PC1      COD     982  1-25-89  10:20a  │ 대응하는 화일이 만들어졌다.
PC1      LST     785  1-25-89  10:20a  ┘
        4 File(s)  16216064 bytes free
```

고 응답 화일이라 불리는 명령어 화일에 의해 필요한 지정을 하는 방식등의 3가지 방법이 있지만, 여기서는 프롬프트 응답방식에 대해서만 설명한다. 그리고 앞에서 설명한 CL 명령어에서는, 내부에서 응답 화일을 작성하여 링커로 건네주어 필요한 정보를 지정하고 있다. 프롬프트 응답방식은 MSC 명령어 기동방법과 마찬가지로 다음과 같은 입력요구가 나온다.

```
Object Modules [.OBJ]
 ;오브젝트 화일명의 입력
Run File [.EXE]
 ;실행가능 화일명의 입력
List File [NUL.MAP]
 ;맵 화일명의 입력
Libraries [.LIB]
 ;라이브러리 화일명의 입력
```

첫번째 프롬프트에 대해 복수의 화일명을 입력하는 경우에는 각각의 화일명 사이를 공백이나 "+"로 구분한다. "Run File"프롬프트에 대해서도 리턴키 만을 누르면 디폴트 입력이 되어 오브젝트 화일명의 베이스명에 ".EXE"를 부가한다. 세번째의 "List File" 프롬프트에서는 맵화일명을 입력한다. 맵화일은 모든 세그먼트에 대한 정보가 들어 있는 텍스트 화일이다. 리턴키 만을 누르면 디폴트의 응답은 "NUL.

〔그림 2〕 CL 명령어에 의한 개발



MAP"이지만, 이것은 리스트 화일을 작성하지 않는 것을 의미한다. 최후의 "Libraries" 프롬프트에서는 라이브러리 화일에 대한 정보를 입력한다. 라이브러리 화일도 오브젝트 화일과 마찬가지로 복수개를 지정할 수 있고, 각각 공백 혹은 "+"로 구분한다. 라인의 최후를 "+"로 마치면 재차 "Libraries" 프롬프트가 표시되고 추가 입력을 기다린다. 화면 4에 링커의 사용예를 보여준다.

4. 라이브러리 매니저(LIB) 라이브러리의 관리

라이브러리 화일이란 필요에 따라 언제라도 인용할 수 있는 오브젝트 모듈의 집합체이다. 프로그램에서 호출하고 있는 라이브러리 함수는, 이 라이브러리 화일에서 LINK에 의해 검색되어 목적으로 하는 오브젝트 모듈만이 추출되어 실행가능 화일에 로드된다. 라이브러리 매니저는, 라이브러리 화일에 대해 다음과 같은 조작을 한다.

〔화면 3〕 CL 명령어의 사용예

```
C>cl pc1.c↵ —— 컴파일과 링크를 모두 실행하려면 확장자 생략 불가.
Microsoft (R) C Compiler Version 4.00
Copyright (C) Microsoft Corp 1984, 1985, 1986.  All rights reserved.

pc1.c —— 컴파일 중임을 나타낸다.
Microsoft (R) Overlay Linker  Version 3.51
Copyright (C) Microsoft Corp 1983, 1984, 1985, 1986.  All rights reserved.

Object Modules [.OBJ]: PC1.OBJ    ┐
Run File [PC1.EXE]: PC1.EXE/NOI   │ 각 옵션에 대해
List File [NUL.MAP]: NUL          │ 디폴트로 수행된다.
Libraries [.LIB]: ;               ┘

C>dir pc1.*↵ —— 무엇이 만들어졌는지 본다.
 Volume in drive C has no label
 Directory of  C:\LANG\MSC4

PC1      C       58  1-25-89   9:25a
PC1      COD    982  1-25-89  10:27a
PC1      LST    785  1-25-89  10:27a
PC1      EXE   6354  1-25-89  10:28a —— 실행가능화일이 만들어졌다.
PC1      OBJ    313  1-25-89  10:28a
        5 File(s)  16211968 bytes free

C>pc1↵ —— 실행해본다.
Hello !
```

〔화면 4〕 링커의 사용예

```
C>link↵ —— 프롬프트 방식으로 해본다.
Microsoft (R) Overlay Linker  Version 3.51
Copyright (C) Microsoft Corp 1983, 1984, 1985, 1986.  All rights reserved.

Object Modules [.OBJ]: pc1↵ —— 확장자 생략 가능
Run File [PC1.EXE]: ↵
List File [NUL.MAP]: ↵
Libraries [.LIB]: ;↵ —— 디폴트로 수행시켰다.

C>dir pc1.exe↵ —— 실행가능한 화일이 만들어졌는지 본다.
 Volume in drive C has no label
 Directory of  C:\LANG\MSC4

PC1      EXE   6354  1-25-89  10:28a —— 실행가능화일이 만들어졌다.
        1 File(s)  16211968 bytes free

C>pc1↵ —— 실행해본다.
Hello !
```

a. 라이브러리 화일의 신규작성
b. 오브젝트 모듈의 삭제
c. 오브젝트 모듈을 오브젝트 화일로써 써낸다.
d. 지정된 오브젝트 모듈을 삭제한다.
e. 지정된 오브젝트 화일을 오브젝트 모듈로서 추가한다.
f. 라이브러리 화일을 다른 라이브러리 화일과 합병한다.
g. 지정된 오브젝트 모듈을 새로운 오브젝트 화일과 치환한다.
h. 라이브러리 화일 중의 모든 글로벌(퍼블릭)심볼의 리스트를 작성한다.
i. 라이브러리 화일내부의 정합성을 검사한다.

LIB의 기동은 LINK와 마찬가지로 프롬프트, 명령어라인, 응답 화일의 3가지 방법을 사용할 수 있지만 여기서는 프롬프트 방법만을 설명한다.

프롬프트에 의한 응답요구는 다음과 같다.

Library Name
:변경 또는 작성할 라이브러리 화일 이름입력
Operation
:조작의 지정
List File
:리스트 화일명의 입력
Output Library
:신규 라이브러리 화일명의 입력

두번째의 "Operation" 프롬프트에서는 라이브러리에 대한 모듈의 추가 및 삭제등의 변경조작을 지정하는 것이고, 그 서식은 다음과 같다.

〈명령기호〉〈이름〉〈명령기호〉〈이름〉[----] [&]

여기서, 〈명령기호〉는 〈이름〉에 대한 조작을 나타내는 일종의 연산자이다. 그리고 〈이름〉은, 오브젝트 모듈명, 오브젝트 화일명, 라이브러리 화일명 중의 하나이다. 또, 마지막의 "&"는 한 라인으로 부족한 경우 연속하여 "Operation" 프롬프트를 받기위한 것이다. 명령기호의 의미와 목적은 표 1과 같다. 세번째의 "List File" 프롬프트에서 임의의 이름을 지정하면 LIB는 라이브러리의 크로스 레퍼런스 리스트를 작성한다. 아무런 지정없이 리턴키를 누르면 리스트 화일은 작성되지 않는다. 최후의 "Output Library" 프롬프트에서는 변경작성된 새로운 라이브러리 화일의 이름을 지정한다. 디폴트는 "Library Name" 프롬프트에서 주어진 이름이다. 또, 이 경우에는 원래의 라이브러리 화일은 확장자가 ".BAK"로 변경되어 보존된다.

실제로 LIB를 사용하여 새로운 라이브러리를 작성하여 본다. LINK할 때, 포함시켜야할 라이브러리가 여러개일 경우 그때마다 모두 타이핑하기란 매우 번거롭다. 그때 여러개의 라이브러리를 하나로 묶어 놓으면 타이핑의 수고도 적고 화일관리도 쉬울 것이다. 그 사용 예를 화면 5에 보였다.

5. MAKE

C의 언어규격이 분할 컴파일 지향적인 면도 있어, C언어에 의한 개발에서는 분할 컴파일 기법이 매우 자주 이용된다. MAKE 명령어는 분할 컴파일을 포함하여 프로그램 개발에 있어 각종의 처리를 반자동화하기 위한 명령어이고, 메이크화일이라 불리는 화일을 준비하는 것에 의해 컴파일을 비롯하여, 링커 및 디버거등 임의의 명령어를 자동적으로 기동시켜, 지정된 화일을 갱신한다. 이때 MAKE 명령어는 각 처리를 할 필요가 있는지 없는지를 판단하는 기능을 갖추고 있기 때문에, 최소한의 필요한 처리만을 하여 작업시간의 낭비를 없앤다.

배치화일에 의한 처리와 MAKE 명령어의 큰 차이점은, 배치화일에 의한 처리가 재처리의 필요성이 없는 것도 포함하여 모든 화일을 처리하는 것에 대하여, MAKE는 미리 정의된 화일의 의존관계에 따라, 화일의 날짜와 시각을 비교하여, 갱신된 필요한 화일만의 재처리를 실행한다고 하는 점이다. MAKE 는 분할 컴파일을 하는 경우 및 어셈블리 언어등의 다른 언어의 프

〔표 1〕 라이브러리 조작을 위한 명령기호와 의미

명령기호	내 용
+	●오브젝트 화일을 그 라이브러리 화일의 최후 모듈로서 추가한다. ● 확장자 ".OBJ"는 생략해도 된다. ●오브젝트화일 대신에 라이브러리 화일의 이름을 지정하면, 그 라이브러리의 내용이 변경되는 라이브러리 화일에 추가된다. 이 경우에는 확장자 ".LIB"를 생략할 수 없다.
-	● 라이브러리 화일에서 모듈을 삭제한다.
'	● 라이브러리 화일중의 모듈을 같은이름의 모듈로 치환한다
*	● 라이브러리 화일중의 모듈을 같은이름의 오브젝트 화일에 복사한다.
-*	● 라이브러리 화일중의 모듈을 같은이름의 오브젝트 화일에 복사한다. 모듈은 라이브러리 화일에서 삭제된다.

로그램과 링크를 하는 경우등, 관리 및 보수해야 하는 원시화일이 증가함에 따라 그 위력을 발휘한다. 특히 디버깅 할 때는 수고와 시간이 크게 절약되어, 효율적인 소프트웨어 개발환경을 제공한다. 다만 한가지 주의해야 할 사항은, MAKE는 그 크기 만큼의 메모리를 별도로 소비하기 때문에 384K 바이트 정도의 메모리에서는 마이크로소프트 C와 함께 사용할 수가 없고, 주기억이 가능하면 640K 바이트까지 확장되어야 한다.

〔화면 5〕 라이브러리 매니저 사용예

```
C>dir *.lib/w↵ —— 라이브러리 화일 리스트를 본다.
 Volume in drive C has no label
 Directory of  C:\LANG\MSC4

SLIBC   LIB  SLIBFP  LIB  EM     LIB  LIBH    LIB
CLIBC   LIB  CLIBFP  LIB  MLIBC  LIB  MLIBFA  LIB
MLIBFP  LIB  CLIBFA  LIB  LLIBC  LIB  LLIBFA  LIB
LLIBFP  LIB  SLIBFA  LIB  87     LIB
      14 File(s)  16209920 bytes free

C>lib↵ —— 라이브러리 매니저를 실행시켜본다.
Microsoft (R) Library Manager  Version 3.04
Copyright (C) Microsoft Corp 1983, 1984, 1985, 1986.  All rights reserved.

Library name: lpc1.lib↵ —— 새로 만들어지는 라이브러리 명을 lpc1.lib로 한다.
Library does not exist.  Create? y↵
Operations: em.lib+libh.lib+llibc.lib+llibfa.lib+llibfp.lib↵
List file: ↵ —— 리스트화일을 만들지 않았다.

C>dir *.lib↵
 Volume in drive C has no label
 Directory of  C:\LANG\MSC4

SLIBC   LIB  101376  6-27-86   4:00a
SLIBFP  LIB   31232  6-27-86   4:00a
SLIBFA  LIB   40960  6-27-86   4:00a
EM      LIB   15872  6-27-86   4:00a ┐
LIBH    LIB   13824  6-27-86   4:00a ┤
CLIBC   LIB  112640  6-27-86   4:00a │
CLIBFP  LIB   31744  6-27-86   4:00a │
MLIBC   LIB  102912  6-27-86   4:00a │
MLIBFA  LIB   41984  6-27-86   4:00a ├─ 이들 5개 라이브러리를
MLIBFP  LIB   31744  6-27-86   4:00a │   모아 LPC1.LIB라는
CLIBFA  LIB   41984  6-27-86   4:00a │   라이브러리로 만들었다.
LLIBC   LIB  114688  6-27-86   4:00a ┤
LLIBFA  LIB   43008  6-27-86   4:00a ┤
LLIBFP  LIB   32768  6-27-86   4:00a ┘
LPC1    LIB  206336  1-25-89  10:51a ←
87      LIB    4096  6-27-86   4:00a
      14 File(s)  16003072 bytes free
```

MAKE를 사용하기 위해서는, 화일 의존관계를 기술한 정의화일(메이크화일)을 작성할 필요가 있다. 화일의 의존관계는 다음과 같이 기술한다.

목적화일(Target) : 관련화일(Source)

또 MAKE 명령어의 사용방법은 다음과 같다.

```
MAKE <메이크화일명>
```

예를 들어, 그림 3과 같은 화일 의존관계는 화면 6과 같이 작성하며, 구체적인 사용예를 화면 6에 보이고 있다. 단, 여기서의 화일 의존관계는 가상한 것이다.

6. 디버거(심볼릭디버거) -SYMDEB

프로그램개발에는 여러가지 장해가 발생한다. 이들은 컴파일때의 에러에서부터, 프로그램이 폭주하는 등의 중대한 버그까지, 각종 경우와 원인이 있다. 이들을 해결하기 위해서는, 우선 컴파일러 및 링커가 출력하는 에러메시지 및 경고메시지를 참고할 수 있지만, 컴파일 및 링크하는 시점에서는 아무런 문제도 발생하지 않았던 프로그램이 프로그램작성자의 의도와 다른 실행결과 및 동작을 야기하는 경우에는 작성자에 의한 시행착오와 함께 심볼릭디버거 등의 디버거를 사용하는 복잡한 디버깅 작업이 요구된다.

컴파일 및 링크 도중에 에러가 나온다든지, 프로그램의 실행이 기대한대로 동작하지 않는 경우에는, 디버그하여 문제가 있는 곳을 정정하고, 재 컴파일 그리고 링크할 필요가 있다. 이러한 경우의 순서를 그림 4에 보인다.

가장 기본적인 디버그 방법으로서는, 의심나는 부위에 임의의 출력메시지를 포함하는 출력문을 삽입하는 방법이 있으나, 이 방법은 프로그램 크기가 크

〔화면 6〕 화일 의존관계 표현 및 MAKE의 사용

```
C>type pcl⏎ —— pcl이라는 메이크 화일의 내용을 본다.

pcl.obj : pcl.c stdio.h —— pcl.c와 stdio.h화일을 사용하며,
          cl -c pcl.c        cl로서 컴파일 만을 수행하여
                             pcl.obj화일을 만든다.
x.obj   : x.c  stdio.h
          cl -c x.c

y.obj   : y.c
          cl -c y.c

z.obj   : z.asm
          masm z.asm;  ——  어셈블리언어로 작성한 프로그램을
                           어셈블하여 오브젝트코드를 생성한다.
pcl.exe : pcl.obj x.obj y.obj z.obj
          cl pcl.obj x.obj y.obj z.obj -o pcl.exe

      └── 오브젝트 화일들을 링크하여 pcl.exe 화일을 만든다.

C>make pcl⏎
make: pcl.obj  -  file not found
cl -c pcl.c
Microsoft (R) C Compiler Version 4.00
Copyright (C) Microsoft Corp 1984, 1985, 1986.  All rights reserved.

make: x.obj  -  file not found
cl -c x.c
Microsoft (R) C Compiler Version 4.00
Copyright (C) Microsoft Corp 1984, 1985, 1986.  All rights reserved.

make: y.obj  -  file not found
cl -c y.c
Microsoft (R) C Compiler Version 4.00
Copyright (C) Microsoft Corp 1984, 1985, 1986.  All rights reserved.

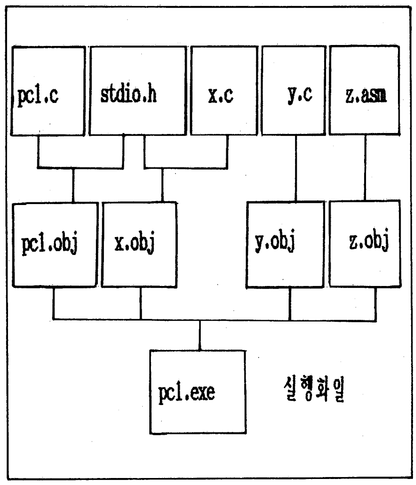
make: z.obj  -  file not found
masm z.asm;
Microsoft (R) Macro Assembler Version 5.00
Copyright (C) Microsoft Corp 1981-1985, 1987.  All rights reserved.

Warning Severe
Errors  Errors
  0       0

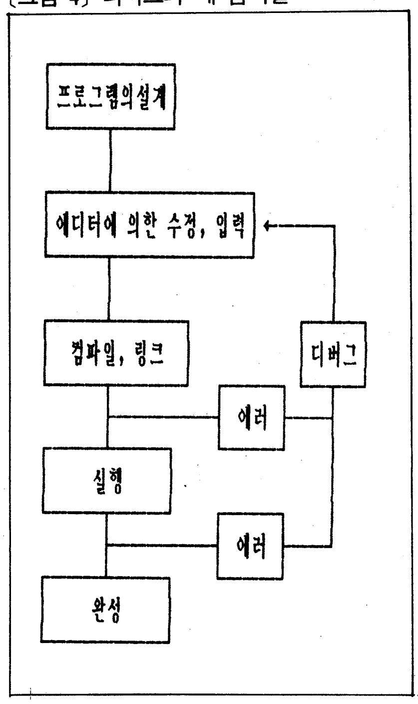
make: pcl.exe  -  file not found
cl pcl.obj x.obj y.obj z.obj -o pcl.exe
Microsoft (R) Overlay Linker  Version 3.51
Copyright (C) Microsoft Corp 1983, 1984, 1985, 1986.  All rights reserved.

C> —— 모든 처리가 끝났다.
```

〔그림 3〕 화일 의존관계



〔그림 4〕 디버그와 재 컴파일



거나 버그의 원인이 복잡한 경우는 버그를 확인, 분석하기가 곤란하다.

심볼릭디버거 SYMDEB는 마이크로소프트 C의 원시화일의 각 라인에 대응하는 디버깅도 할 수 있는 범용 디버거이다. SYMDEB의 가장 큰 특징은 기계어 코드 뿐만 아니라, C언어 등 고급언어의 원시 프로그램을 취급

할 수 있다는 것이다. 예를 들어 프로그램 중의 특정위치의 참조는, 어드레스 이외에 그로벌 심볼 및 원시 프로그램의 행번호에 의해 수행할 수 있다. 프로그램의 표시도 어셈블리 언어, 원시 프로그램 코드 및 양자의 조합으로 수행할 수 있지만, C 언어에 의한 프로그램의 디버그를 SYMDEB에 의해 수행하는 경우에는 로컬변수를 그 이름에 의해 참조할 수 없기 때문에 스택의 내용을 참조하도록 해야 한다. SYMDEB의 사용방법은 다음과 같다. 보다 자세한 내용은 기회를 보아 다음으로 미룬다.

```
SYMDEB <심볼화일명> <실행가능화일명> [<인수>]
```

7. 그외의 유틸리티

(1) EXEMOD

이 유틸리티는 실행가능 화일의 EXE 헤더(화일헤더)를 변경하는 경우 선두에 위치하여, 재배치 정보, 스택포인터의 초기값, 메모리의 패러그래프 크기, 실행가능 화일의 크기 등의 정보를 갖고 있다. EXEMOD 유틸리티는 이중에서 스택포인터의 초기값과 메모리의 패러그래프 크기를 옵션을 사용하여 변경할 수 있다. 사용하는 서식은 다음과 같으며 화면 7은 실행 예이다.

```
EXEMOD <실행가능화일명>
[-stack<스택크기>][-min<최소크기>][-max<최대크기>]
```

(2) EXEPACK

이 유틸리티는 지정된 실행가능 화일 중 동일 분자배열(배열 및 구조체를 초기화하기 위한 NULL문자)를 압축하여, 재배치 표를 최적화한다. 그 결과로서, 실행가능 화일의 크기가 작아져, 프로그램의 로드에 필요한 시간이 단축된다. 서식은 다음과 같다.

〔화면 7〕 EXEMOD 사용예

```
C>exemod pc1.exe↵
Microsoft (R) EXE File Header Utility  Version 4.00
Copyright (C) Microsoft Corp 1985.  All rights reserved.

pc1.exe                         (hex)          (dec)

.EXE size (bytes)               18D2           6354
Minimum load size (bytes)       2302           8962
Overlay number                     0              0
Initial CS:IP                0000:0039
Initial SS:SP                01B0:0800         2048 → 스택크기
Minimum allocation (para)         C3            195 → 최소크기
Maximum allocation (para)       FFFF          65535
Header size (para)                20             32
Relocation table offset           1E             30
Relocation entries                 3              3
Relocation entries                 3              3
```

〔화면 8〕 EXEPACK 사용예

```
C>exepack pc1.exe pc11.exe↵
Microsoft (R) EXE File Compression Utility  Version 4.03
Copyright (C) Microsoft Corp 1985.  All rights reserved.

C>dir pc1?.exe↵
 Volume in drive C has no label
 Directory of  C:\LANG\MSC4

PC1      EXE     6354   1-25-89  10:37a
PC11     EXE     6333   1-25-89  10:53a  — 실행가능화일 pc1.exe를 pc11.exe로 압축하였다.
         2 File(s)  15994880 bytes free
C>
```

〔화면 9〕 MAPSYM의 사용예

```
C>mapsym pc1.map↵  —  심볼릭디버깅을 위해 심볼화일을 만든다.
Microsoft (R) Symbol File Generator  Version 4.00
Copyright (C) Microsoft Corp 1984, 1985.  All rights reserved.

        Program entry point at 0000:0039
C>
```

```
EXEPACK <실행가능화일명> <새 화일명>
```

EXEPACK에 의해 수정된 화일은 〈새화일명〉에 다시 써 넣으므로, 원래의 화일은 변경되지 않으며, **화면 8**에 실행예를 보인다.

(3) MAPSYM

LINK할 때 만들어진 맵화일(.MAP)로부터 심볼화일(.SYM)을 작성한다. 이 심볼화일에는 글로벌 변수의 어드레스 정보 등이 들어 있어, SYMDEB로 원시프로그램 코드 수준의 디버그를 할 경우에 필요하다. 서식은 다음과 같고, **화면 9**에 실행예를 보인다. 이렇게 하면, SYMDEB를 사용하기 위한 준비가 끝나게 된다.

```
MAPSYS <맵 화일명>
```

사용자 인터페이스 규격

프로그램을 사용하는 사용자에게는 명령어 사용의 편리성을 판단하는 요소는 그렇게 많지 않다. 프로그램의 실행속도 및 기능은 말할 것도 없지만, 지원도구로서 사용하는 명령어등의 경우에는 자신이 사용하는 명령어의 통일감이 중요한 요소가 된다. 예를 들면 옵션의 지정, 이용할 때 준비해야할 필요가 있는 화일, 설정하는 환경 등이 명령어마다 다르다면 아무리 처리속도가 빠르고, 기능이 풍부해도 그 명령어의 매력은 반감할 것이다. 이와같이 프로그램의 규격을 결정할 때는 그 프로그램의 실행환경, 즉 MS-DOS의 규격을 무시할 수 없다. 여기서는 MS-DOS용으로 작성된 프로그램에 대해 그 사용자 인터페이스 규격을 결정할 때 주의해야 할 사항에 대해 설명한다.

1. 명령어라인의 형식

명령어라인에 있어서 옵션 및 처리화일의 지정은, 사용자에게 가장 가까운 명령어 인터페이스이기 때문에' 명령어의 서식은 가장 중요한 사용자 인터페이스라고 할 수 있다. 또 명령어에서 인수의 해석은 명령어의 기능구현과 함께 프로그램에서의 기둥부분이라 말할 수 있을것이다. 그래서 MS-DOS의 명령어라인 서식에 대해 알아본다.

MS-DOS와 함께 제공되는 표준 명령어의 대부분은, CP / M의 명령어를 MS-DOS용으로 고쳐 만든 것이고 다음과 같은 성질을 갖고 있다.

a. 각 인수는 "," 및 공백으로 구분.
b. 각 옵션은 " / "뒤에 지정한다.
　옵션을 지정하는 장소는 명령어에 따라 다르다.
c. 처리하는 화일의 수는 고정됨.

첫번째의 인수 구분방법은 운영체계의 구조와 밀접하게 관련되는 것이다. MS-DOS의 경우 명령어라인은 리다이렉트 및 파이프의 지정을 제외하고는 그대로 명령어로 넘기기 때문에 구분문자는 명령어마다 자유로이 결정할 수 있다. 그래서 MASM(매크로어셈블러)등의 명령어는 구분문자로서 ","를 사용하고 있고, COPY등의 명령어는 공백을 사용하고 있어, 명령어의 규격에 통일감이 없다. MS-DOS상의 명령어라도, C 언어를 이용하는 경우에는 프로그램의 초기화루틴(컴파일러 및 링커에 의해 부가된다)이 명령어라인을 해석하고, 공백으로 구분되는 것은 별도의 인수로서 분해하므로 C 언어로 명령어를 작성하는 경우에는 필연적으로 공백을 구분문자로서 택하게 된다. 또 어셈블리 언어에 의해 개발하는 경우, 구분문자를 자유로 선택할 수 있지만, 명령어의 통일감을 갖게 하기 위해서는 공백을 선택하는 것이 좋을 것이다.

MS-DOS의 대부분의 명령어에서, 옵션은 " / "의 뒤에 하나밖에 쓸 수 없다. 또 LINK의 옵션처럼 각 옵션이 2문자 이상으로 표현되는 경우도 많아 옵션이 장황하다는 느낌이 든다. 특히 작성자 이외의 사용자가 사용하는 경우에, 이 명령어의 규격은 한번 결정하면 변경하기가 곤란하고, 사용자가 스스로, 사용하기 편리하게 변경하는 것도 번거롭고 어렵기 때문에 프로그램의 작성자는 이러한 것까지 고려하여 규격을 결정한다.

또, MS-DOS의 대부분의 명령어는 처리하는 화일의 갯수가 대체로 한개이고, MORE 및 SORT 명령어등은 화일을 지정하는 것조차 불가능하여 리다이렉트를 이용하여야 한다. 그러나 사용의 용이성 면에서 보면 명령어의 성질상 화일 갯수가 한정되는 것 이외에는 임의 갯수의 화일이름을 지정할 수 있도록 하는 것이 좋을 것이다. 예를 들어 DEL 명령어를 보자. MS-DOS의 경우 인수를 와일드문자("*" 및 "?")를 사용하는 것 외에는, 화일명은 하나 밖에 지정할 수 없다. 2개 이상의 화일을 주려할 때는 2회 이상의 DEL 명령어를 실행시켜야 한다. 따라서 가능한 한 복수의 화일을 지정할 수 있도록 지원하는 것이 바람직하다고 할 수 있다. 본 연재중에서는 이러한 명령어를 C 언어로 작성하고 실행 및 사용방법에 대해 설명하기로 한다.

2. 환경의 이용

버전 2.0 이후의 MS-DOS에서는 "환경"이라는 개념이 들어 있다. 환경은 SET 명령어에 의해 정의할 수 있고, 또 배치처리에서 빈번히 이용되지만, 이 환경변수의 값은 임의의 프로그램 중에서도 이용가능하여, 범용의 명령어를 작성하는 경우에는 기종 및 시스템구성에 의존하는 부분을 환경으로 정의하여 프로그램의 이용범위를 넓힐 수 있다.

3. 제어화일

앞에서 설명하겠지만 MS-DOS상의

LINK 및 LIB의 응답화일 그리고 MAKE 명령어의 메이크 화일등은 제어화일로서, 옵션 및 환경에 의한 지정만으로는 명령어를 제어할 수 없기 때문에 명령어를 제어하기 위해 별도의 화일을 준비하고, 명령어는 그 내용을 읽어들여 처리한다. 즉, 명령어를 제어하기 위한 데이터화일을 "제어화일"이라 부른다. 제어화일을 이용하는 것에 따라, 명령어는 소위 "프로그래머블"하게 되고, 그 명령의 기능도 거의 무제한으로 확장할 수 있게 된다. 그러나 한편으로는 명령어의 기동때마다 제어화일을 읽어야 하므로 명령어의 실행속도는 상당히 제한을 받게 된다. 그러므로 바꿔말하면, 제어화일을 이용하는 명령어는 속도를 희생하여 고도의 기능을 제공하는 것이다.

4. 쉘 이스케이프(Shell Escape)

대화형의 소프트웨어, 예를 들면 에디터 및 멀티플랜등을 이용하고 있을 때, 그 소프트웨어의 내부에서 다른 명령어를 기동하고 싶은 경우가 있다. 에디터의 내부에서 DIR 명령어를 기동한다든지, 멀티플랜을 이용하여 일을 수행하면서 PRINT 명령어로 다른 화일을 인쇄한다든지 하는 것은 자주 경험했던 것일 것이다. 이와같은 요구에 부응하기 위해 프로그램의 내부 명령어를 증가시키는 방법은 2가지면에서 문제가 발생한다. 하나는 프로그램 구조 및 규모가 복잡, 비대해지고, 보수가 매우 어렵게 된다는 것과 또 하나는 같은 기능을 가진 명령어가 별도로 준비되어 있는 데도 그 기능을 이용하지 않는 것이다.

쉘 이스케이프란 "다른 명령어들을 수행한다"고 하는 기능이지만, 이것을 준비하는 것에 의해 프로그램을 효율적으로 사용할 수 있다.

5. 도큐먼트(Document)

모든 프로그램은 매뉴얼 등의 도큐먼트가 있어야 "도구"로서 이용할 수 있는 것이고, 도큐먼트가 없어서는 그 이용가치 및 활용도가 반감한다. 매뉴얼은 명령어의 용도 및 대상으로 하는 사용자에 따라 다르지만 표준적인 매뉴얼에는 다음과 같은 항목을 기술하고 있다.

a. 명령어의 이름과 기능 설명.
b. 명령라인의 서식
c. 옵션의 해설
d. 기능의 상세한 설명
e. 명령어가 사용하는 화일(제어화일 등)의 이름과 설명
f. 관련된 명령어 루틴류의 열거
g. 명령어의 이용시 혼동하기 쉬운점의 주의사항
h. 사용상의 주의사항;
 -실현하지 못하는 기능, 규격의 제한, 미해결버그 등
i. 에러 메시지의 설명
j. 사용 예, 참고문헌 등

6. 기타

그외의 규격으로는 출력 디바이스의 선택과, 명령어가 출력하는 메시지가 있다. 또한 MS-DOS가 원칙적으로 영숫자만을 표현할 수 있는 ASCII 코드를 채택하고 있으므로 한글코드의 대응도 생각하면 좋을 것이다.

프로그램 작성시의 내부 규격

여기서는 MS-DOS 및 하드웨어와 밀접하게 관련된 것에 대해 언급한다.

1. 입출력 데이터의 버퍼링

대부분의 명령어, 특히 필터 명령어에서는 처리시간의 대부분을 입출력하는데 소비한다. 따라서 명령어의 속도를 향상시키는 데는 입출력의 속도를 높이는 것이 가장 좋다. 입출력 데이터를 프로그램에서 버퍼링하면, 프로그램의 구조는 다소 복잡하게 되지만, 처리속도는 크게 향상하기 때문에, 많은 입출력 프로그램에서는 필수적인 기법이다. 버퍼링에 대한 상세한 설명은 본 강좌의 제5회에서 설명한 것이다.

2. 환경

어떤 프로세서에서 자 프로세서를 기동하는 경우, 자 프로세서의 환경을 자신의 프로세서와는 전혀 다른 내용으로 할 수 있지만, 그런 경우는 드물고, 자기자신의 환경을 그대로 자 프로세서에 넘기는 것이 일반적이다. 그래서 최상위의 프로세서(대개는 COMMAND. COM)에서 정의된 환경이 최후의 프로세서까지 넘겨져, 환경을 참조하는 프로그램을 효과적으로 작성할 수 있게 된다.

3. 임시화일의 이용

임시화일(Temporary File)은, 그 명령어가 실행되고 있는 동안에만 존재하는 화일이므로, 그 존재는 일반사용자에게는 보이지 않는다. 따라서 임시화일명이 기존의 화일과 같지 않도록 주의를 기울여야 한다. 또 그것은 명령어가 종료시에는 삭제되는 것이지만 치명적인 에러가 발생한다든지 사용자가 CTRL-C의 입력에 의해 처리를 종료한다든지 할 때는 임시화일은 삭제하는 것이 바람직하다. 그리고 그것은 디스크상에 작성되므로 그 디스크 장치의 속도가 그대로 명령어의 실행속도에 반영된다. 따라서 임시화일은 가능한 한 램디스크와 같은 고속디시크 장치상에 작성하는 것이 바람직하며 그 실현은 환경변수 TMP를 정의하여 두는 것으로 가능하다.

4. 예외처리와 출력메시지

모든 소프트웨어(명령어) 개발에 있어서, 예외처리는 사용자 인터페이스 중에서 큰 비율을 차지하고 있다. 출력하려는 디스크가 꽉차있는 경우, 명령

어라인의 지정이 올바르지 않는 경우, 또 사용자가 CTRL-C를 눌러 프로세스를 종료시킨 경우 등, 소프트웨어(명령어) 자신에게 버그가 없더라도 예외처리는 항상 존재한다. 예외처리의 주안점은, 소프트웨어(명령어) 처리 속행의 여부, 적절한 메시지의 출력, 그리고 제외 원인에 대하여 어떤 대처를 할 것 인가 하는 것이다.

예외가 발생할 가능성은 대체로 다음과 같은 것이다.

a. 프로그램의 실수 및 처리를 완전히 실행하지 못하는 조건이 발생한 경우
b. 명령어라인 및 제어화일의 지정이 올바르지 않은 경우
c. 입력데이터(처리 데이터)의 내용 및 양식이 바르지 않는 경우
d. 입출력등의 시스템 콜할 때 발생하는 경우
e. 치명적인 에러
f. CTRL-C에 의한 사용자의 실행중지 요구

일반적으로, 발생할 수 있는 에러를 모두 검출하는 것은 대단히 어려운 것이다. 또, 그와 동시에 프로그램크기의 증가, 실행속도의 저하, 프로그램의 복잡화 등 여러가지 부작용이 생긴다. 따라서 프로그램을 작성할 때, 어떤 에러를 검출할 것인가 하는 판단은, "명령어 실행에 지장을 주는지 어떤지"를 가늠으로 하여 결정하면 좋다. C 처리계에서는 "signal()"이라는 라이브러리 함수를 이용하는 것에 의해, 예외처리를 하는 함수를 지정할 수 있다.

보통, 에러가 있으면 그 상황에 맞는 메시지를 출력하게 된다. 그 내용은 간결하고, 적절하게 함축되어야 한다. 기본적으로는 다음과 같은 요소가 포함되어 있는 것이 바람직하다.

a. 에러가 발생한 명령어의 이름
b. 에러의 내용(메모리 부족, 디스크 꽉 참 등)
c. 무시, 치리 중단 등의 에러처리 내용

그리고 에러의 성질에 따라서는, 메시지 출력후 바로 프로그램 처리를 종료하게 되지만, 이 경우, 이 이후에 실행되는 프로세스에 대하여 시스템의 모드 변경등에 의한 영향을 주지 않도록 보통 다음과 같은 처리를 한다.

a. 임시화일 삭제
b. 버퍼에 남아있는 데이터 처리
c. 인터럽트 벡터의 복원 및 이용하고 있던 인터럽트의 금지
d. 레코드록 해제
e. 화면에 대한 설정의 복원
f. 각종 모드의 복원
g. 에러 코드의 넘김

5. 프로그램의 범용성

MS-DOS가 제공하는 평션 콜 등의 기능을 이용하는 것에 따라, 기종에 의존하지 않고 이식성이 높은 프로그램을 작성할 수 있다. 그런데 MS-DOS에서는, DOS가 이용할 수 있는 환경을 넓히기 위해, 개개의 하드웨어에 의존하는 부분에 대해서는, 일체 지원(표준화)하고 있지 않다. 그 대표적인 예로서, 마이크로소프트 C의 그래픽 기능을 들 수 있다. 현재 대부분의 기종에서는 그래픽 기능을 구현할 수 있지만, MS-DOS에서는 그래픽 기능에 대해 전혀 지원하고 있지 않는다. 이것은, 다른 자원(메모리 및 디스크 장치)과 비교하면, 기종마다 해상도 및 표시할 수 있는 색의 종류 등 그래픽장치의 기능의 차가 너무 커서, DOS가 그 차이를 완전히 흡수할 수 없기 때문이다. 이 결과 프고그램에서 그래픽 기능을 사용하려면, 각각의 기종에 의존하지 않을 수 없다.

지면관계상 지원도구에 대한 자세한 내용과 프로그램에서 고려해야할 사항들을 세세히 설명하지 못 하였지만 앞으로 기회가 닿는대로 보충 설명해 나갈 예정이다. 특히, 컴파일러 드라이버 및 링커의 충분한 활용은 프로그램 개발효율 및 특정 목적용으로 매우 중요한 지원도구이다. 그들의 원활한 사용을 위해 옵션지정에 관한 내용으로, 초보자 중심의 설명을 위하여 화면중심의 설명을 하는 별도의 기회를 만들어 설명할 것이다. 다음 회에서는 프로그램 개발시 MS-DOS와 프로그램 사이의 각종 데이터 교환을 하는 영역인 PSP(Program Segment Prefix) 및 시스템콜, 그리고 C 언어와 다른 언어, 특히 어셈블리 언어 프로그램과의 링크에 대해 설명하겠다.

마이크로프로세서 마이크로컴퓨터 (하드웨어 과정) 講義案內

개강 매월 5일 (2개월 완성)

본원에서는 금번 2개월 과정으로 digital circuit 설계로 부터 CPU 및 memory, 주변장치 등의 구조 및 동작원리를 습득하여 computer system 및 주변기기의 Interface를 설계하는 능력을 함양하고자 합니다. 또 이 과정에서는 Z-80계 computer 및 APPLE, IBM PC 등의 system을 대상으로 구성, 동작원리, 프로그래밍 실습을 하여 궁극적으로 MICOM 및 digital system의 설계 및 수리능력을 배양하고자 하오니 많은 참여 바랍니다.

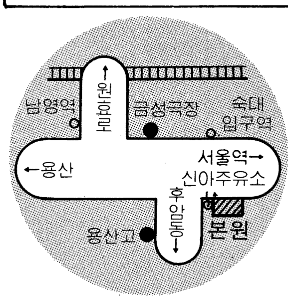
■ 강습일정

주별	일차	강습내용 (이론 및 실기)
1주	1일차	computer의 기본 구조와 8bit CPU의 종류 및 내부 구성
	2일차	논리회로 설계원리 및 그 응용
	3일차	마이크로 컴퓨터 시스템의 구성
2주	1일차	Karnaugh Map을 이용한 디지탈회로 설계
	2일차	Flip Flop 및 Counter 등을 이용한 응용회로 설계
	3일차	CPU의 주변회로 및 Address Map/Decoding
3주	1일차	TTL 논리회로 설계 실습
	2일차	A/D 및 D/A Converter 및 ALU
	3일차	Memory(ROM/RAM/DRAM)의 구성
4주	1일차	8080 및 8085 CPU의 구조 및 Timing 특성
	2일차	8085 Assembly언어 프로그래밍
	3일차	I/O(Port, Keyboard etc)의 원리 및 Video Generator
5주	1일차	Z80 CPU의 구조 및 Timing 특성
	2일차	Z80 CPU의 Interrupt (IM0, IM1, IM2)
	3일차	μ-Computer의 회로설계 및 실습
6주	1일차	Assembly Language를 이용한 프로그램 실습
	2일차	8255 및 Z80 P10의 특성 및 응용
	3일차	16bit MPU(MC68000)의 구조 및 원리
7주	1일차	Z80 S10의 특성 및 응용
	2일차	EPROM Writer 및 Floppy Disk Controller
	3일차	MC68000의 Signal 및 Bus Operation
8주	1일차	음성합성 장치의 설계
	2일차	Bus Standard, Modem, Data통신 소개
	3일차	16bit MPU(MC68000)의 회로 설계

Textbook

1. TTL응용실무
2. Z80 마이크로컴퓨터
3. Microprocessor Interfacing Technique
4. Microprocessor and Digital Systems

Hardware + Software = ?



전자산업 4반세기를 지켜온 기술인의 전당

과인 한국전자통신학원 컴퓨터기술부

(구 한국전파학원)

140 서울시 용산구 갈월동 17번지 ☎ 754-1751/2902/1044

마이크로프로세서의 구성

김용득/아주대 전자과 교수

서론

전자공학의 응용은 크게 제어분야와 정보처리분야로 나누어진다. 여기에는 단순한 제어장치의 응용에서부터 로보트, 항공 전자공학, 군사용 장비, 복잡한 제어장치의 응용과 특수목적의 정보처리 분야 등 그 활용분야는 매우 다양하다 하겠다. 기본적인 응용에서는 종래의 전자회로를 사용하나, 복잡한 시스템에서는 이들 회로를 사용하는 것이 경제적이지 못하기 때문에 복잡한 회로를 표준화한 프로그램형 반도체 소자들이 개발되어 생산되고 있다.

특히 최근의 반도체 설계방식은 하나의 하드웨어 소자에 특정조건을 인가하면 다양한 기능을 수행하는 이른바 프로그램형 LSI가 보편적이고 이들 중 널리 알려진 소자를 마이크로프로세서라 부른다. 즉 마이크로프로세서는 컴퓨터의 핵심부분인 중앙연산처리장치(CPU)를 하나의 반도체 소자로 만든 것으로 연산, 판단, 처리 등의 기능을 수행할 수 있다.

최근에 사용되는 반도체 소자의 집적기술은 VLSI*방식으로서 하나의 반도체내에 16비트 프로세서 및 주변소자로 조합된 시스템장치를 단일 회로내에 구성시켜 사용하기 쉽도록 만들고 있다. 현재 프로세서분야의 주요 제조회사는 자일로그, 모토롤라, 인텔사 등이며, 여기에서는 인텔의 8086 / 286 및 그 주변장치 등에 대하여 다음과 같은 방식으로 설명하고자 한다.

즉, 본 강좌에서는 마이크로프로세서 시스템을 목적에 따른 효율적인 설계방법과 이의 활용방법에 관하여 수회에 걸쳐 연재하고자 한다.

제1부에서는 마이크로프로세서 내부 동작특성을 이해하기 위하여 프로세서의 제어장치를 중점적으로 설명하고자 한다. 이는 현재 널리 사용되고 있는 다양한 프로세서를 자유자재로 활용하기 위한 필수적 요소라 본다.

2부에서는 가장 보편적인 16비트프로세서를 사용한 하드웨어 설계방법을 연구하고, 3부에서는 명령어의 구성형식과 이를 활용한 시스템 응용방식에 중점을 두어 설명하고, 4부에서는 시스템 동작을 위한 시스템 프로그램 개념과 소형 마이크로프로세서 시스템을 구성하고, 이의 모니터 프로그램을 작성하여 작동시켜 보도록 한다.

5, 6부에서는 마이크로프로세서의 주변기기 접속방식과 정보교환방식에 관하여 기술하고 7부에서는 기억장치 구성과 동작개념을, 8부에서는 멀티프로세서 시스템에 관하여 설명하고자 한다.

마이크로프로세서 시스템의 구성

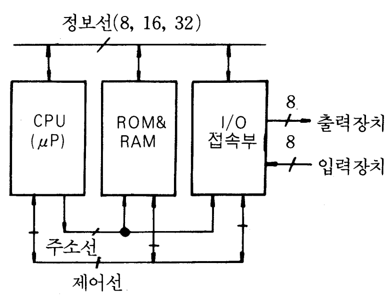
마이크로프로세서 시스템은 다른 컴퓨터와 마찬가지로 하드웨어, 소프트웨어와 휨웨어를 포함한다. 하드웨어란 회로의 소자와 이의 구성방식을 포함한 전자회로방식의 구성체제이고, 소프트웨어란 주어진 일을 효율적으로 컴퓨터 내에서 처리하기 위한 명령어의 집합들로 컴퓨터 사용자들의 처리언어를 구성하는 것이며, 휨웨어는 하드웨어의 성능을 향상시키고 사용자의 편리성을 강조하기 위한 하드웨어 보조용 프로그램이라 볼 수 있다.

마이크로프로세서 시스템의 하드웨어 구성은 주요 처리기능을 갖는 CPU와 기억소자 및 입·출력 장치 등을 상호연결한 것으로 그림 1과 같다.

여기서 CPU란 마이크로프로세서로서 주어진 명령을 해석하고 시스템에서의 작동을 제어하며, 모든 연산과 논리적인 계산을 수행한다.

기억소자(ROM과 RAM)는 프로세서에서 사용되는 명령문과 정보를 저장하는 역할을 하며 이들은 보통 몇개의 모듈(IC)로 되어 있고, 각 모듈은 수천개 이상의 기억장소를 가지고 있다. 따라서 각 기억장소를 지정하기 위하여는 주소번지를 사용하여 지정하게 되며, CPU는 이들 주소번지를 할당하여 정보를 기억, 또는 읽어 처리하게 된다.

〔그림 1〕 마이크로프로세서 시스템



입·출력 장치는 외부와의 연락과 많은 양의 정보를 표시하고자 할때 사용한다.

그러면 이들 마이크로프로세서 시스템에서 정보가 어떻게 처리되는가를 이해하기 위하여 먼저 마이크로프로세서 내부에서 정보가 어떻게 처리되는가에 관하여 설명하기로 한다.

정보처리방법

마이크로프로세서의 내부 회로구성은 연산처리부*와 레지스터부* 및 제어부*로 구성되어 있고 그 기능은 매 순간마다 필요한 정보를 이동하는 기능과 특정 정보의 내용을 변형하는 기능으로 나누어 볼 수 있다.

그러나 컴퓨터의 대부로 알려진 Von Neumann박사에 의한 컴퓨터 정의는 반드시 처리될 순서가 기억장치에 이미 내장되어 이로부터 연속적으로 읽어 처리하여야 한다는 것이다. 따라서 이를 위한 명령어 구성형식은

① OP코드
② 첫번째 정보군
③ 두번째 정보군
④ 결과가 기억될 주소번지
⑤ 다음 처리될 명령어의 주소

등이 포함되어야 하며, 이의 하드웨어 구성을 간편하게 하기 위하여 다음에 처리될 명령어의 주소는 항상 PC(Program Counter)에 있도록 하며, 결과가 저장될 위치도 Acc*(Accumulator)라는 특정 내부레지스터로 지정하고 처리될 정보의 주소값도 특정 레지스터에 할당하면 매번 처리될 명령어의 형식이 간소화될 것이다.

물론 처리될 다른 하나의 정보주소는 주소지정방식(Addressing Mode)에 의하여 다음호에 언급하겠지만, 처리될 정보의 내용을 어떻게 효율적으로 처리하는가를 논하는 것이며, 결국 단위시간에 필요한 정보를 빠르게 저장하기 위해서는 내부레지스터가 많을수록 유리하고, 이로 인하여 처리할 명령어 표시방법이 간편하게 됨을 알 수 있다.

〔표 1〕 명령어 표

명령어	기계어 코드	설명
MOVE	9C−−	3바이트 명령어, OP코드 다음의 내용을 DC에 기억
LOAD	40	1바이트 명령어, DC에서 지정한 메모리 내용을 Acc에 저장
ADD	80	1바이트 명령어, Acc내용과 DC에서 지정한 기억장소값과 더하여 Acc에 저장
STORE	60	1바이트 명령어, Acc의 내용을 DC에서 지정한 기억소자에 저장

그렇다면, 이들 CPU 내부 레지스터가 왜 필요하며, 어떻게 명령어 처리 과정중에 사용되어지는가를 쉽게 이해하기 위하여 몇개의 레지스터를 지정하여 설명하겠다. 앞에서도 설명하였듯이 Acc는 모든 연산 처리중 필요한 정보값과 연산처리된 후에 결과가 기억되기 때문에 처리될 정보가 8비트라면 8비트 레지스터가 되고, 16비트 정보라면 16비트 레지스터가 될 것이다. 즉, 이들 정보길이에 따라 8비트 또는 16비트 프로세서로 구분된다고 볼 수 있다.

다음 처리될 정보는 기억소자에 있다고 하고 이의 총수가 64K개라면 16개 주소선으로 지정가능하며, DC(Data Counter)에 의하여 이들 기억소자의 정보저장번지를 지정한다. 따라서 16비트가 필요하게 됨을 알 수 있다.

또한 어떠한 종류의 명령어를 수행하는가를 OP 코드에 의하여 지정하고, 이를 처리하기 위하여 일정 시간이 필요하게 된다. 그러므로 이들 명령어를 처리하는 동안 OP코드를 기억시킬 IR*(Instruction Register)이라는 레지스터가 필요하고, 256개의 다른 종류의 명령어가 있다면 이들 IR은 8비트가 요구된다.

PC는 현재의 명령어가 처리된 후 다음 명령어를 처리하기 위한 위치가 기억된 주소값을 기억하게 되므로 기억소자의 총용량값을 지정할 수 있어야 하므로 16비트가 필요하게 된다.

그러면 이들 4개의 CPU내부 레지스터 즉 Acc, IR, DC, PC가 명령어 처리 과정중 어떻게 사용되는가를 "C=A+B"를 예로 들어 수행시키면서 설명하고자 한다.

먼저 이를 수행하기 위하여 표 1과 같은 4개의 명령어를 가정하자. 표 1에서 보면 명령어는 1바이트 또는 3바이트로 구성되나 반드시 첫 바이트는 OP코드부로서 무엇을 할 것인가를 지정하게 된다. 이들 명령어를 사용하여 "C=A+B"를 처리하기 위한 기억장소에 저장된 처리순서는 표 2와 같이 편의상 기억소자의 400번지부터 기억되어 있다고 가정하고 A정보 주소는 "0A30", B정보 주소는 "0A31"에 기억되어 있으며 결과인 C정보 주소를 "0A31"번지에 기억시키고자 한다.

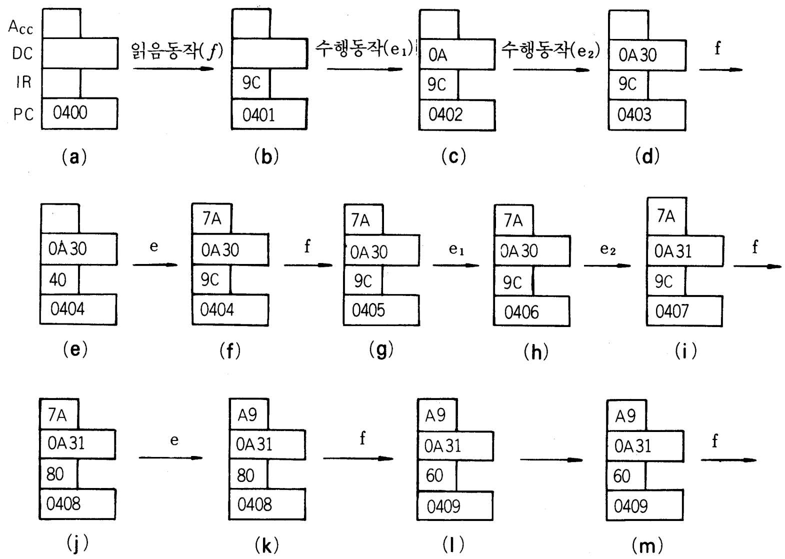
이들 명령어의 처리는 CPU내부에서 마이크로 오퍼레이션이라는 무수한 세부동작으로 이루어지는데 이들의 공통항으로 하드웨어 설계상에서 보면 명령어 읽음동작(Fetch)과 이를 수행하는 동작(Execute)으로 이루어지므로 단위 일을 처리하기 위한 프로그램 형태로 보면 읽음과 수행동작의 반복으로 볼 수 있다. 따라서 이를 세부동작으로 설명하면 그림 2와 같다.

그림 2의 (a)는 앞의 명령어가 처리된 직후 내부 레지스터 상태로서 Acc, IR, DC에는 임의 값이 기억되어 있지만, 고려할 필요가 없는 상태이고, 지금 처리할 명령어가 PC의 상태로서 400번지에 있음을 의미하게 된다. 따라서 명령어 읽음동작이 이루어지면 그림 2의 (b)와 같이 OP코드는 IR 레지스터에 기억되고, PC는 다음 처리할 명령어를 지시하도록 1이 증가한다. OP

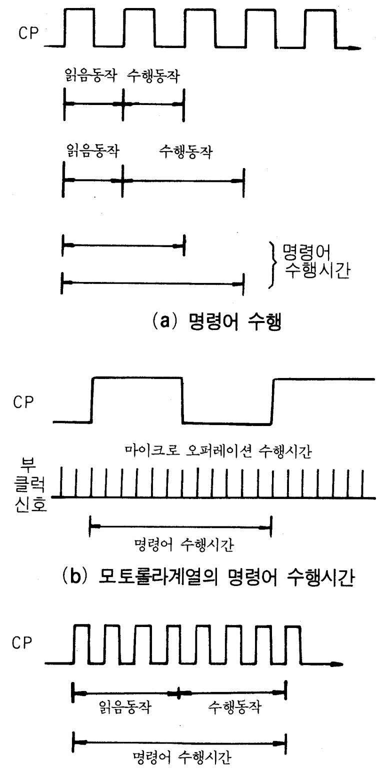
〔표 2〕 지정된 프로그램내용

주소구분	기억소자의 주소값	정보내용	설　　명
프로그램 기억소자부	0400	9C	명령어 1, DC에 0A30 저장
	0401	0A	
	0402	30	
	0403	40	명령어 2, Acc←〔M〕DC
	0404	9C	명령어 3, DC←0A31
	0405	0A	
	0406	31	
	0407	80	명령어 4, Acc←〔Acc〕+〔M〕DC
	0408	60	명령어 5, 〔M〕DC←〔Acc〕
	⋮	⋮	
정보 기억소자부	0A30	7A	0408번지 수행후에는
	0A31	2F	A9가 기억됨

〔그림 2〕 명령어 처리과정



〔그림 3〕 명령어 수행시간



(a) 명령어 수행

(b) 모토롤라계열의 명령어 수행시간

(c) 인텔계열의 명령어 수행시간

코드 읽음동작에서 IR이 9C이면 이의 처리과정에서는 PC에 지정된 연속된 기억소자로부터 값을 읽어 DC에 저장하라는 의미가 되고 이는 (c), (d)와 같이 2개의 처리 수행시간이 필요하게 된다. 이때 동작지정은 마이크로프로세서를 어떻게 설계하느냐에 따라 제조회사에 의하여 결정된다.

이들 수행동작이 끝나면 404번지에 있는 다음 명령어를 읽어 IR에 40을 기억시키고, 처리하게 된다. 즉, OP코드가 40을 DC에서 지정한 기억소자번지로부터 읽어 Acc에 기억시키라는 의미이므로 수행동작이 끝나면 Acc에 7A가 저장된다. 이와 같이 매 클럭이 인가될 때마다 PC에 지정된 번지로부터 명령어를 읽어 처리하게 되므로 독자 스스로 그림 2를 추적하며 내부 레지스터와 기억소자 변화값을 이해하길 바란다.

명령어 수행시간

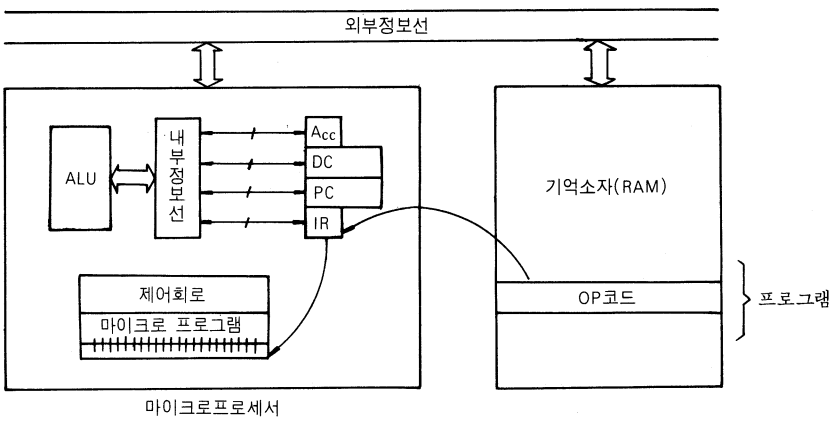
앞절에서는 명령어가 마이크로프로세서 내부에서 어떻게 수행되는가를 설명하였으나 본 절에서는 이들 명령어들이 처리되는 수행시간에 관하여 논의하겠다. 이는 마이크로프로세서 시스템의 하드웨어 설계상에서 매우 중요한 것이며 다음호에 설계될 입·출력 접속부에서 필수적으로 활용될 것이다.

프로세서에서 명령어가 특정시간에 동작을 일치시켜 모든 내부회로가 처리되길 위하여는 외부적으로 클럭펄스라는 시간 제어신호를 인가하여야 한다.

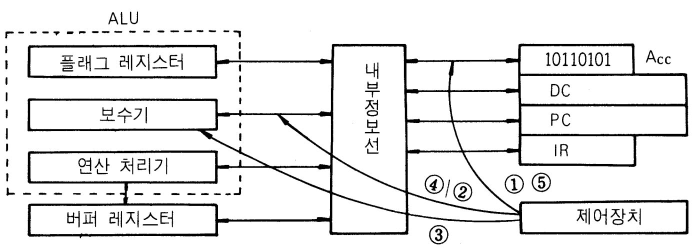
이들 클럭펄스는 **그림 3**과 같은 일정 주기의 구형파로 CP(Clock Pulse)라는 기호를 사용하며 CP에 따라 명령어 읽음동작과 수행동작이 이루어지게 된다. 특히 수행동작은 앞의 예제에서 보여준 것과 같이 1개 또는 2개의 CP가 필요하고, 이들을 명령어 수행시간이라 한다. 그러나 이들 외부클럭에 의

한 동작은 마이크로프로세서 제조회사에 따라 조금씩 다르나 그림 3(b)와 같이 대분될 수 있다. 즉 모토롤라계열의 프로세서는 하나의 클럭이 내부적으로 여러개의 부클럭(Sub Clock Pulse) 으로 나누어져 이들 부클럭이 회로내부의 마이크로 오퍼레이션을 수행하게 되며, 또 인텔계열 프로세서는 인가된 외부클럭을 기본시간(State Time)으로 하여 프로세서의 각 내부회로를 작동시키게 되므로 결과적으로 여러개의 외부클럭이 하나의 명령어 수행시간이 된다.

〔그림 4〕 마이크로프로세서 제어부 동작



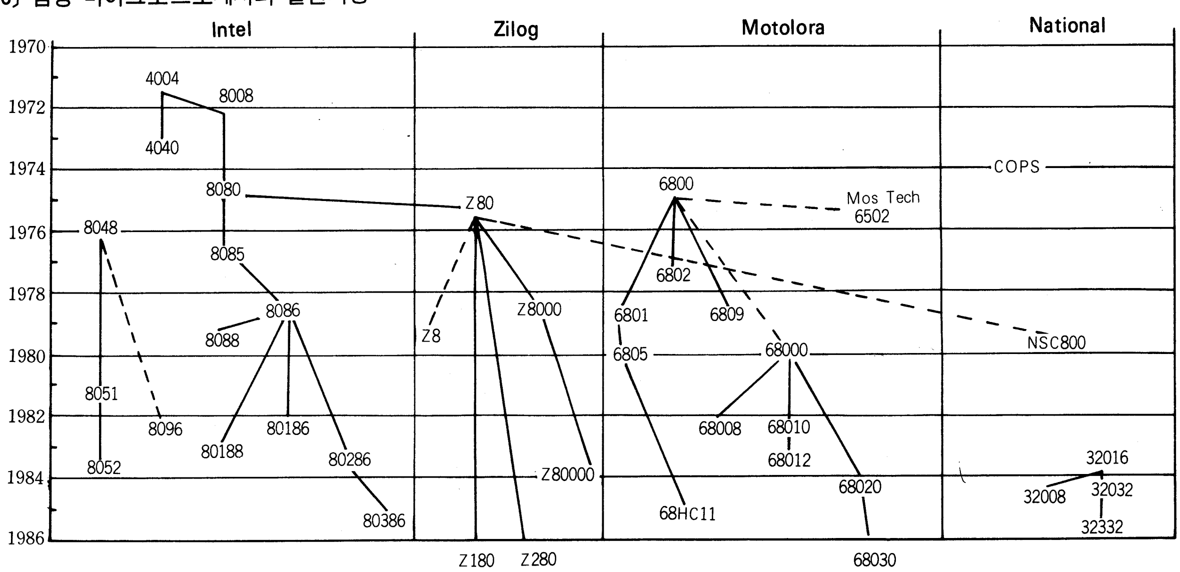
〔그림 5〕 마이크로프로그래밍 수행순서



마이크로프로세서 시스템은 **그림 4**와 같이 구성되는데 이의 작동을 보면 RAM에 처리될 명령어군 즉 프로그램이 기억되어 있다. 이는 PC에 기억된 주소값에 의하여 처리될 명령어가 읽혀지면, OP 코드부는 IR에 기억되며 IR에 기억된 코드는 프로세서 제어부의 OP 코드 디코더에 의하여 해석되어 제어회로부의 특정 ROM 내용이 읽혀져 처리하게 된다. 이들 제어회로부내의 특정 ROM 내용은 마이크로프로그램이라 하며, 모든 명령어를 세부적으로 구분하여 내부 회로소자의 작동신호를 구동하게 된다. 이들 과정을 이해하기 위하여 Acc의 내용을 보수를 취하는 (Acc) 명령어 수행과정에 대한 예제를 들어 설명하겠다.

그림 5와 같이 Acc에 기억된 수행전 값은 "10110101"이며, 처음 수행동작은 클럭펄스에 의하여 Acc의 내용이

〔그림 6〕 범용 마이크로프로세서의 발전과정



내부 정보선에 이동되고, 다음 클럭에서는 연산장치의 보수기에 이 값이 전달된다. 3번째 동작은 연산 장치의 보수기가 작동되어 "10110101"이란 정보값이 보수를 취하여 "01001010"의 값이 되고 역순으로 Acc에 전달된다.

즉 외부에서 볼때, 그림 3에서 하나의 클럭에 의하여 수행된 과정이 그림 5와 같이 5개의 세부 동작으로 이루어지게 되며, 이러한 순서가 제어부의 ROM에 내장되어 있다. 이는 제조회사에서 명령어를 설정할 때 이미 구성되어져 있으므로 마이크로프로세서 사용자는 실제로 마이크로프로그램을 응용하지는 않게 된다.

이 외에도 비트 슬라이스 프로세서(Bit Sliced Microprocessor)가 있는데 이들 마이크로프로그램을 사용자들이 직접 구성해야 하므로 좀더 구체적인 비트 슬라이스 프로세서에 대한 연구는 독자 스스로 습득하길 바란다.

표준 마이크로프로세서

마이크로프로세서 구성은 앞에서 설명한 바와 같이 연산처리부, 내부레지스터부 및 제어처리부로 나누어져 있다. 또한, 제조 회사별로 특징을 갖고 하나의 LSI 반도체칩으로 상품화되어 있으므로 우리는 이들의 외부 단자를 이해하고, 다른 소자와 접속하여 시스템을 구성하게 된다.

따라서 이들 범용화된 프로세서의 외부단자 연결법을 본 강좌에서 연재하므로 다양한 기능의 마이크로프로세서 활용방법을 습득하길 바란다.

이들 범용 프로세서의 발전과정은 그림 6과 같이 1971년 인텔의 4004를 시작으로 75년경에 인텔 8080, 자일로그사의 Z80, 모토롤라사의 M6800 등 8비트가 출현하여 본격적인 프로세서 시스템이 산업적 응용에 활용되기 시작하였고, 최근에는 16비트, 32비트 및 단일 칩 마이크로컴퓨터의 등장으로 가전기기에까지 적용되고 있다. 이들 모든 범용 프로세서의 구성은 본란에서 설명된 기본적인 구성으로 이루어졌고, 본 강좌에서는 이들 상품화된 범용 마이크로프로세서 중 현재 IBM-PC라 불리는 시스템에 적용되어 가장많이 보급된 인텔계열의 16비트 프로세서를 사용한 소형컴퓨터를 중심으로 시스템구성과 활용에 관하여 다음호부터 기술하기로 하겠다. PC

〈용어해설〉

●VLSI
LSI보다 집적도가 높고, 한개의 칩에 1만개 이상의 트랜지스터 소자를 집적한 IC이다. 20만개 이상의 트랜지스터를 갖는 VLSI도 있으며 초LSI 또는 초대규모(초고밀도) 집적회로라고도 한다.

●ALU(Arithmetic and Logic Unit)
CPU내의 장치의 하나로서 2진수 데이터의 산술연산, 논리연산을 행한다. 일반적으로 어큐뮬레이터의 내용과 메모리의 내용 또는 레지스터와 레지스터의 내용이 연산되고, 결과는 어큐뮬레이터 또는 레지스터에 남는다. 산술연산에는 덧셈, 뺄셈, 곱셈, 나눗셈이 있고, 논리연산에는 AND, OR, EXOR 등이 있다.

●레지스터(Register)
1비트 또는 복수의 비트를 유지하는 기억장치나 회로로서 지정의 목적에 사용되며 수시로 그 내용물을 이용할 수 있도록 되어 있다. CPU 내부에는 여러개의 레지스터가 있으며 각종의 명령을 실행할 때 명령의 내용 또는 데이터를 일시적으로 유지하는데 사용된다. 또한 레지스터는 플립플롭에 의해서 구성되어 있다.

●CU(Control Unit)
제어부에서는 레지스터와 메모리 장치에 필요한 제어변수를 만들어 낸다. 시스템에서는 24개의 제어변수가 있으며, 이들을 제어함수로 나타낼 수 있다.

●PC(Program Counter)
어드레스를 저장하는 레지스터로 일반적으로 명령이 메인 메모리로부터 읽어서 실행단계에 들어가면 프로그램카운터의 내용에 1이 더해진다. 이 때문에 CPU는 프로그램 카운터가 나타내는 어드레스 명령을 메인 메모리로부터 순차적으로 읽어 실행할 수 있다.

●Acc(Accumulator)
연산장치에 있는 중요한 레지스터로서 사칙연산, 논리연산 등의 결과를 저장하는 것이다. 보통 하나의 수치가 저장되어 있으며 다른 수치가 들어오면 양자의 대수합으로 바뀐다. 또, 저장된 수치에 대해서는 자리보내기, 보수 등의 조작도 할 수 있는 것이 보통이다.

●IR
CPU내에 있는 제어 레지스터의 하나로서 메모리에서 읽어낸 명령 코드를 받아서 이것을 실행하기 위해 일시 기억시켜 두는 레지스터이다.

●보수(Complement)
고정기수 표기법에 있어서 주어진 수를 어떤 특정한 수에서 빼서 얻어지는 수로서, 기수를 b로 했을 때 주어진 수에서 다음의 방법에 의해 유도된다.
① b에 대한 보수(b's complement) : 주어진 수의 각 자리의 수를 b-1에서 빼고 최하위의 1을 더하여 필요하면 자리올림을 한다.
② (b-1)에 대한 보수((b-1)'s Complement) : 주어진 수의 각자리의 수를 b-1에서 빼고 최하위에 1을 더하여 필요하면 자리올림을 한다.

●메모리 어드레스 레지스터(Memory Address Regzter)
중앙처리장치에 있는 레지스터로서 리드나 라이트를 하는 메모리의 위치(어드레스)를 기억하는 소용량의 일시기억회로.

●메모리 셀(Memory Cell)
메모리의 가장 작은 부분으로 그속에 데이터의 1단위를 넣어 두거나 보존할 수 있고, 또 그것을 꺼낼 수도 있다.

●원칩 마이크로 컴퓨터(One Chip Microcomputer)
하나의 칩에 CPU, PROM, RAM, I/O버스, 타이머, 카운터, 클럭 등을 수용한 마이크로컴퓨터. 풍부한 기능과 메모리 또는 I/O의 확장성이 높으며, 지금까지의 마이크로 컴퓨터보다 소형, 저가격으로 시스템을 구성할 수 있다.

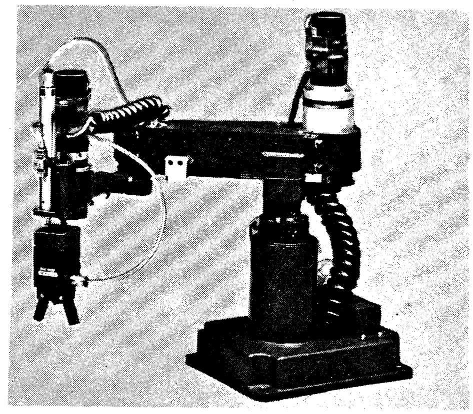
●동기(Synchronization)
시스템의 구성요소가 클럭신호에 맞춰서 동작하는 것 즉, 데이터 전송에 있어서 데이터 전송속도를 맞추는 등 약속에 따라 각 엘리먼트 또는 엘리먼트군의 시간관계를 송수 상호간에서 일치시키는 것.

●심볼 어드레스(Symbol address)
프로그래밍을 할 때 어드레스로서 인간이 알기 쉬운 적당한 기호를 붙여서 나타내는 것. 그 기호 어드레스가 절대 어드레스의 어디에 상당하는 가는 시스템이 판단하므로 프로그래머의 부담이 적어진다. 따라서 어셈블러언어의 프로그래밍에서는 거의 이 기호 어드레스를 사용하고 있다.

마이크로프로세서를 이용한
FA SYSTEM과정
(생산자동화)

※ 개강 :
매월 5일

講義案內 (2개월 완성)

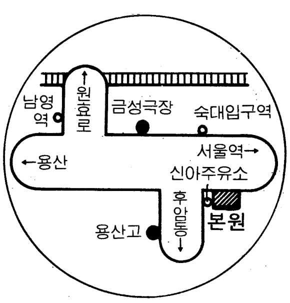


본원에서는 다년간 Computer Hardware 과정을 교육하여 많은 수료생을 배출, 각계에서 공헌하고 있는 바 금번 생산업계의 요청에 따라 컴퓨터 프로세서를 이용한 생산체제와 산업용 로보트의 발전으로 제품과 제조공정의 설계 및 계획, 공장 현장에 있는 생산 제조장치의 컴퓨터 제어, 원료 및 자재의 유동제어, 공간 활용 및 작업계획 등의 조직적 기획 등의 필요성에 따른 FA 과정의 이론, 실기를 위주로 특강을 하게 되었으니 관계 직원의 많은 참여 바랍니다.

■ **강의일정**

주별	일자 \ 과정	강습내용 (이론 및 실기)	주별	일자 \ 과정	강습내용 (이론 및 실기)
1주	1일차	Logic회로 설계의 기초	5주	1일차	PWM System
	2일차	Micom System		2일차	Power Control System
	3일차	KKH 80A H/W구조		3일차	DC Motor Control
2주	1일차	Z80 Assembly 언어	6주	1일차	Step Motor의 기초 이론
	2일차	KKH 80A S/W 구조		2일차	Step Motor Control
	3일차	PIO 입출력 실습		3일차	Senser Interfacing(1)
3주	1일차	A/D Converter의 제작	7주	1일차	IBM PC H/W 구조
	2일차	D/A Converter의 제작		2일차	IBM PC/XT Interfacing
	3일차	DC Motor Drive회로의 설계		3일차	RS 232C Serial Interfacing
4주	1일차	Monitor Program의 설계	8주	1일차	Senser Interfacing(2)
	2일차	응용 Program 설계(1)		2일차	Micro Robot(1)
	3일차	응용 Program 설계(2)		3일차	Micro Robot(2)

전자산업 4반세기를 지켜온 기술인의 전당



과인 **한국전자통신학원** 컴퓨터 기술부

(구 한국전파학원)

140 서울시 용산구 갈월동 17번지 ☎ 754-1751/2902/1044

순서논리회로

김정선 /한국항공대학 전자과 교수

플립플롭 회로

2개의 능동소자를 연결하여 한쪽이 전도상태(ON)일 때 다른 한쪽이 차단상태(OFF)가 되도록한 일종의 재생회로(Regenerative Circuit)를 멀티바이브레이터(Multivibrator)라고 한다.

이러한 멀티바이브레이터의 기본적인 회로에는 쌍안정(Bistable), 단안정(Monostable), 무안정(Astable) 멀티바이브레이터가 있다. 이들 중에서 쌍안정 멀티바이브레이터를 플립플롭(Flip Flop)이라고 한다.

플립플롭의 종류는 플립플롭이 가지는 입력의 수와 입력이 플립플롭상태에 영향을 미치는 방법에 따라 여러가지로 나뉘는데 주로 사용되는 플립플롭으로서는 RS 플립플롭, D 플립플롭, T 플립플롭, JK 플립플롭 등이 있다.

플립플롭회로는 주로 기억회로, 카운터, 레지스터 등에 응용된다. 디지탈 시스템에 있어서 순서논리회로의 기억회로는 **그림 1**에서와 같이 조합회로 출력의 일부를 기억하였다가 조합회로의 새로운 입력과 함께 조합회로에 입력되므로 다음의 순서회로 출력을 결정하는 일을 한다.

(1) RS 플립플롭

기본적인 RS 플립플롭 회로는 2개의 NAND 게이트나 2개의 NOR 게이트로 구성된다. **그림 2**는 2개의 NOR 게이트로 구성한 RS 플립플롭이며, 입력이 S단자와 R단자에 가해져서 출력 Q와 Q′가 결정된다. 출력은 입력이 S단자와 R단자에 가해지는 것에 따라서 세트상태 또는 리세트상태로 된다. 여기서 세트상태는 출력Q가 1상태("참" 출력 또는 높은 전압상태)이고, 출력Q가 0상태("거짓" 출력 또는 낮은 전압상태)인 것을 말한다.

즉, 세트상태일 때는 Q=1, Q′=0이고, 리세트상태일 때는 Q=0, Q′=1이다.

그림 2에서는 입력R이 0상태이고 입력S가 1상태일 때 출력은 세트상태로 되고, 입력R이 1상태이고 입력S가 0상태일 때 출력은 리세트상태가 된다.

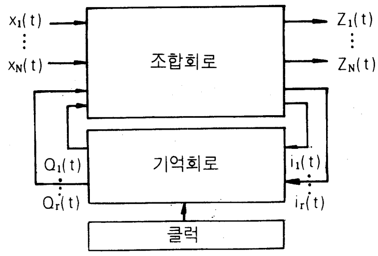
그러나 S=R=0인 상태에서는 그림 2(b)의 진리표에서 볼 수 있는 바와 같이 출력상태는 변하지 않고 현재 상태 그대로 유지된다. S=R=1인 상태에서는 출력Q와 Q′는 모호한 상태로 결정적인 상태가 못되기 때문에 이러한 입력은 RS 플립에서는 피하는 것이 좋다.

그림 2(a)는 비동기 RS 플립플롭이지만 그림 3(a)와 같이 2입력 AND게이트에 클럭 펄스(Clock Pulse:CP)를 가하게 되면 클럭 펄스에 동기가 된 RS 플립플롭이 된다. 이렇게 클럭된(Clocked) RS 플립플롭에서는 CP=1인 경우에만 R과 S의 상태에 따라 출력 Q가 변화되고 CP=0인 경우에는 R과 S의 상태는 출력 Q에 영향을 주지 않는다.

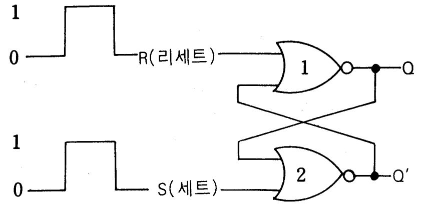
그림 3(c)는 현입력 S, R과 현출력 Q에 대한 다음의 출력 Q(t+1)의 상태를 나타낸 진리표이고, 그림 3(d)는 (c)의 진리표로 작성한 카르노맵(Karnaugh Map)과 특성방정식을 표현한 것이다.

그림 4는 클럭된 RS 플립플롭을 이용한 일예로서 워드타임신호(Word Time Signal)를 발생시키는 회로이다. 이 회로는 처음에는 3비트 카운터를 0으

〔그림 1〕 순서회로의 블록다이어그램



〔그림 2〕 NOR 게이트를 사용한 RS 플립플롭



(a) 논리회로

(S=0, R=0일 때 전출력 그대로 유지)

S	R	Q	Q′
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0

(b) 진리표

로 클리어시킨다. 시작신호가 플립플롭 Q를 세트시키면 이 플립플롭의 출력이 워드타임제어를 1로 만들고 카운터를 인에이블 시킨다. 이렇게 8개의 펄스가 공급된후에 플립플롭은 리세트되어 Q는 0이 된다.

(2) D 플립플롭

D 플립플롭은 그림 5(a)와같이 1개의 입력클럭(CP)과 1개의 입력데이터(D)를 갖는 플립플롭 회로이다.

이 D 플립플롭 회로는 클럭펄스가 인가되면 반전작용없이 입력D의 논리신호가 플립플롭 출력에 나타난다. 즉, 입력D의 신호가 0이든 1이든간에 클럭펄스가 나타나지 않는한 출력에는 아무런 영향을 주지 못한다.

D 플립플롭은 클럭펄스에 의하여 입력D의 신호가 출력측에 그대로 전송되므로 클럭펄스만큼 시간지연하게 된다.

RS플립플롭을 사용하여 D플립플롭을 구성하려면, S입력은 직접 연결하고 R입력은 S입력을 반전(NOT)시켜서 입력시키면 된다. 그림 5(a)는 2개의 NAND 게이트를 사용해서 구성한 RS 플립플롭의 S입력에 D입력을 직접 연결하고, 하나의 NAND 게이트를 통해서 D입력을 R입력에 가해서 구성한 D 플립플롭 회로이다.

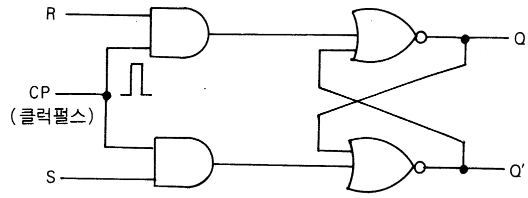
그림 6은 D 플립플롭을 이용하여 구성한 2진식 리플카운터이고, 3단 상향 리플카운터로서 각단의 출력 Q를 2단의 입력D로 되돌려 연결하여 반전플립플롭으로 동작토록한 카운터이다.

(3) JK 플립플롭

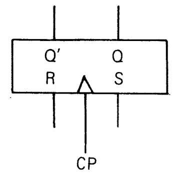
JK 플립플롭은 디지탈시스템에서 가장 널리 사용되고 있는 플립플롭 회로이다. 그림 7에서와 같이 입력단자 J(Jack)와 K(King), 클럭펄스 입력단자 CP, 출력단자 Q와 Q′를 갖고 있다.

JK 플립플롭을 구성하려면 RS 플립플롭에다 약간의 논리게이트를 추가해서 구성할 수도 있고, 그림 7과 같이 그림 3(a)의 RS 플립플롭으로부터 출

〔그림 3〕 클럭된 RS 플립플롭



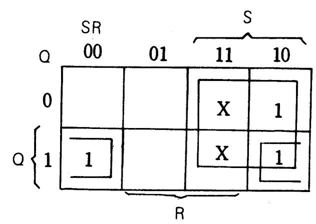
(a) 논리회로



(b) 기호

Q	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	모호한 상태
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	모호한 상태

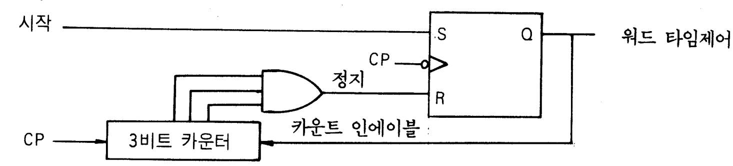
(c) 진리표



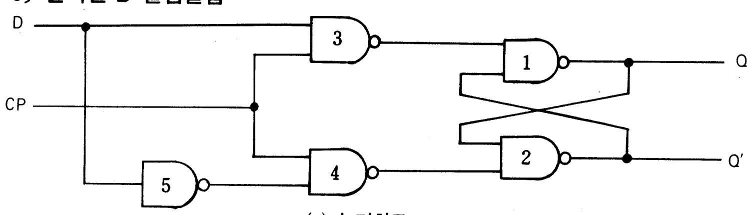
Q (t+1)=S+R′Q SR=0

(d) 특성방정식

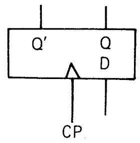
〔그림 4〕 워드타임신호 발생기회로



〔그림 5〕 클럭된 D 플립플롭



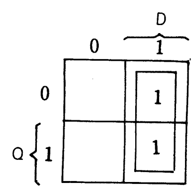
(a) 논리회로



(b) 기호

Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

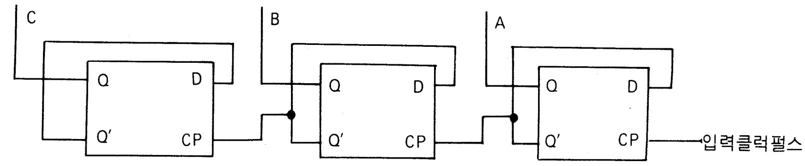
(c) 진리표



Q(t+1)=D

(d) 특성방정식

〔그림 6〕 D 플립플롭을 이용한 3단 상향 리플카운터



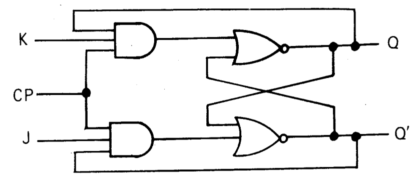
력Q와 Q′를 각각 K와 J로 궤한시켜서 구성할 수도 있다.

JK 플립플롭의 가장 중요한 특징은 RS 플립플롭에서 모호한 상태를 얻어 금지시켰던 S=R=1인 입력을 가능하게 한 점이다. 다시말하면 J=K=1인 입력을 가하면 클럭펄스에 의해서 반전작용이 생긴다.

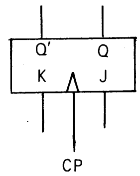
그림 7의 회로에서 JK 플립플롭의 동작을 살펴 보면 J=K=0인 경우에는 현출력의 상태를 그대로 유지하고, J=1, K=0인 경우에는 세트상태(Q(t+1)=1)로 된다. 그리고 J=0, K=1인 경우에는 리세트상태(Q(t+1)=0)이며 J=K=1인 경우에는 다음 출력은 현출력상태의 반전(Toggle)된 값을 갖게 된다.

그림 8은 JK 플립플롭을 이용해서 구성한 좌측 시프트레지스터이다. 각단은 JK 플립플롭으로 구성되어 있고 입력J와 K에는 서로 반대되는 2진식 신호가 인가되며, 시프트신호가 각단에 공급되면 동시에 동작이 일어나서 J와 K단자의 데이터입력은 해당되는 레지스터단으로 옮겨가고 시프트펄스가 생겼을 때의 입력데이터에 대응해 각단을 세트나 리세트상태로 만든다.

〔그림 7〕 JK 플립플롭



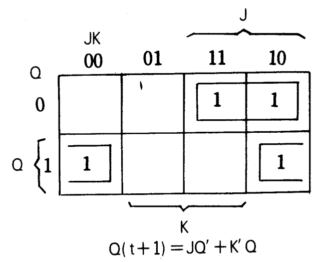
(a) 논리회로



(b) 기호

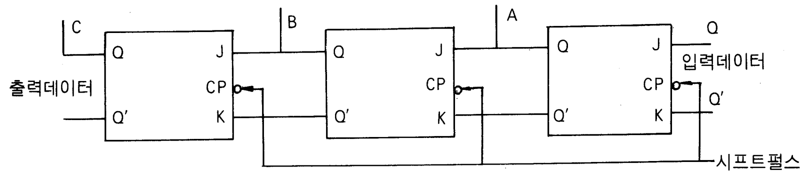
Q	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(c) 진리표



(d) 특성방정식

〔그림 8〕 JK 플립플롭을 이용한 좌측 시프트 레지스터



(4) T 플립플롭

T 플립플롭은 **그림 9**와 같이 한개의 클럭입력펄스(CP)와 출력Q, Q′를 갖는 회로로서 각 클럭입력펄스에 따라 출력의 상태가 변하는 반전플립플롭(Toggle Flip Flop)회로이다.

JK 플립플롭에서 J=K=0일 때 현재의 출력상태를 유지하는 특성과 J=K=1 일 때 현재의 출력상태를 다음의 출력상태로 반전시키는 특성을 이용하여 T 플립플롭을 구성할 수 있다. 즉, 그림 9와 같이 JK 플립플롭의 JK입력단자를 묶어서 T 입력으로 사용하면 된다.

그림 10은 T 플립플롭을 사용하여 구성한 기본적인 2단 2진카운터이다. 이 카운터는 클럭입력에 따라 0에서 3까지 카운트할 수 있는 회로로서 제1단의 출력Q가 제2단의 반전입력 T에 연결되어 있으므로 제1단의 출력Q가 제2단을 반전(트리거)하고, 제2단의 출력은 제1단의 출력Q가 높은 전압상태에서 낮은 전압상태로 변할 때마다 변화한다.

(5) 플립플롭의 레이스문제

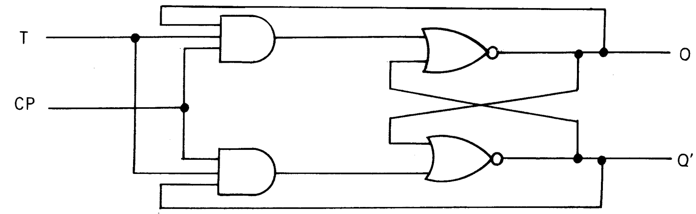
그림 7(a)에서 J=K=1이고 Q=0일 때 CP=1이면 2개의 게이트를 통과하는 시간 △t 만큼 지연된 후에 출력Q는 1로 된다. 즉 J=K=1, Q=1이 되고, 클럭지속시간(Duration Time)을 t_p라고 하면 t_p시간동안은 클럭CP가 계속 1의 상태를 유지하므로 출력Q는 입력측으로 궤환되어 계속해서 0과 1의 상태를 반복하게 된다. 이런 반복작용은 t_p가 끝날 때까지 계속되며 t_p가 끝난 후 출력Q는 0이나 1중 어떤 상태로 될 것인지는 불확실하게 된다.

따라서 이와 같은 출력측의 상태변화로 입력측이 변화되어 오동작이 발생되는데 이런 현상을 레이스(Race)현상이라고 한다.

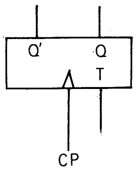
레이스현상을 제거하려면 클럭의 지속시간 t_p를 게이트 전달지연시간 △t보다 작게 해서 한 입력에 의해서 결정된 출력이 J=K=1일 때 입력측으로 궤환이 되지 않게 해 주면 된다.

즉, $t_p < \triangle t < T$(T는 클럭주기)가 되는 첨예한 펄스(Sharp Pulse)를 사용하면 된다. **그림 11**은 클럭펄스를 표시한 것이다. 다음은 레이스현상을 일으키지 않는 회로를 예를 들어 설명한 것이다.

〔그림 9〕 T 플립플롭



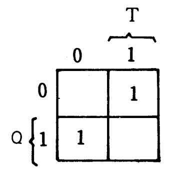
(a) 논리회로



(b) 기호

Q	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

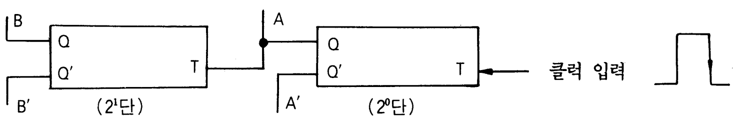
(c) 진리표



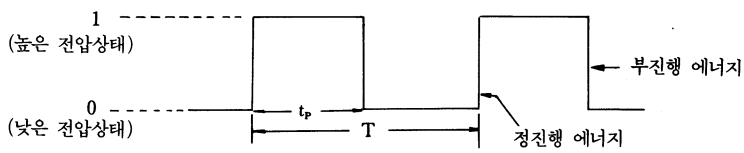
$Q(t+1) = TQ' + T'Q$

(d) 특성방정식

〔그림 10〕 기본적인 2단 2진 카운터



〔그림 11〕 클럭 펄스



●에지 트리거 플립플롭

레이스현상이 없도록 플립플롭을 구성하는 방법으로서 에지 트리거 플립플롭(Edge-triggered Flip Flop)을 이용하는데 주로 RTL 집적회로로 만들어 진다.

그림 12에서 AC입력회로인 전단의 S입력에 1이 CP의 부진행에지에서 가해지면 그림 12에서처럼 래치(Latch) 입력에 부 펄스가 발생한다.

이때 S와 클럭 CP에 방형파가 가해졌을 때 NAND 게이트 하나를 거치는 지연시간에 상당하는 만큼의 짧은 부펄스가 래치입력에 발생한다. 이 지연시간은 수nsec 정도로, 레이스문제를 해결한다.

●주종 플립플롭

주종 플립플롭(Master Slave Flip Flop)회로는 레이스 문제를 없애기 위해서 가장 많이 쓰이는 방식으로서 DTL과 TTL회로에 모두 쓰인다. 이 플립플롭은 주 플립플롭과 종 플립플롭으로 구분되고 CP=1인 동안 입력S와 R이 주 플립플롭에 가해지더라도 종 플립플롭의 클럭입력이 NOT 게이트를 거쳐서 가해지므로 주 플립플롭의 출력이 종 플립플롭에 전달되지 못하고 CP=0이 될 때 비로서 종 플립플롭에 전달되어 출력에 나타나게 된다. 따라서 주종 플립플롭이 클럭펄스에 의하여 분리되므로 레이스현상으로 인한 오동작이 발생하지 않는다.

그림 13은 주종 플립플롭의 논리회로의 일예이다.

카운터

카운터(Counter)는 컴퓨터를 비롯하여 수를 계수(카운트)하는 응용분야에서 널리 사용되는 것으로서 클럭입력에 의하여 동작할때 수나 상태의 시퀀스를 통해서 나타낼 수 있는 논리회로이다.

또 카운터는 임의코드를 순환 반복할 수 있으며, 설계방법에 따라 원하는 코드와 순환순서를 결정할 수 있다.

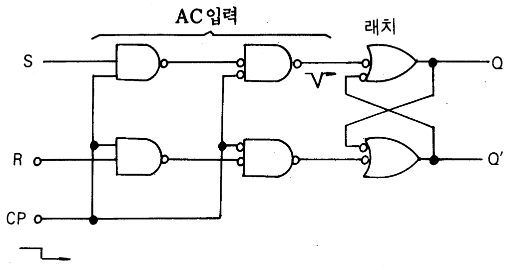
일반적으로 카운터는 동기식카운터와 비동기식카운터로 구분되고, 이들은 대부분 플립플롭으로 구성된 기억회로와 논리게이트로 구성된 조합논리회로로 구성된다. 우리는 기억회로와 조합논리회로로 구성된 그림 1과 같은 회로를 순서논리회로라고 한다. 디지탈 논리회로에서 조합논리회로는 입력의 변화에 따라 출력이 결정되고 과거의 동작 상태에는 관계하지 않는 회로이다. 그러나 순서논리회로는 이러한 조합논리회로에다 기억회로를 포함하고 있으므로 입력의 변화 뿐만 아니라 기억회로의 현상태 즉, 과거의 동작상태에 의하여 출력이 결정된다.

기억회로의 입력은 조합논리회로의 출력의 일부가 되고, 기억회로의 출력은 조합논리회로의 새로운 입력과 함께 가해져서 순서논리회로의 다음상태를 결정한다. 카운터를 설명하기 전에 순서논리회로에 대해서 좀더 구체적으로 살펴보기 위해서 순서논리회로의 설계과정을 살펴보기로 한다.

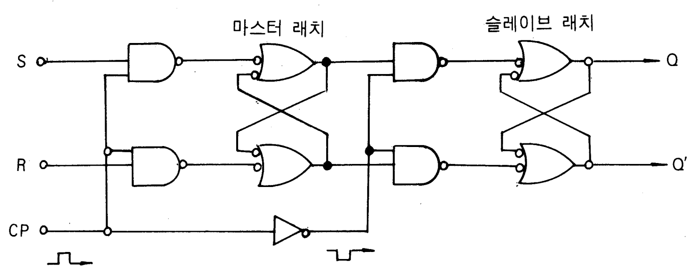
(1) 순서논리 회로 및 설계과정

그림 1에서 입력변수($X_1(t),\cdots,X_N(t)$), 출력변수($Z_1(t),\cdots,X_N(t)$) 기억회로의 현상태 ($Q_1(t),\cdots,Q_r(t)$) 등의 설명을 쉽게 하기 위해서 입력변수를 X, 출력변수를 Z, 기억회로의 현상태를 Q(t)라고 하자.

〔그림 12〕 에지 트리거 플립플롭



〔그림 13〕 주종 플립플롭회로



동작원리는 앞서서 말한 바와 같이 입력변수와 기억회로의 현상태로서 출력변수가 결정되고 동시에 기억회로의 다음 상태가 기대된다. 따라서 같은 입력값을 주더라도 기억회로의 현상태가 다르게 되면 출력값이 달라질 수 있게 되어 조합논리회로만의 경우와는 달리 해석이 다소 까다롭다.

순서논리회로의 동작설명을 위해서 위해서 그림 14와 같은 상태천이도를 생각해 보자.

각 상태는 특정한 상태표시를 나타내는 원으로(각원 내부에 있는 8은 그 원이 나타내는 상태를 표시한다.) 화살표는 입력 X의 각 조건에 대해서 어떤 상태로 그 회로가 가게 되는가(즉, 상태천이)를 표시한다. 그림 14는 4개의 상태를 표시하므로 순서논리회로의 기억회로는 최소한 2개의 플립플롭이 필요하다 (1개의 플립플롭으로서 2개의 상태 즉, Q(t)=0, Q(t)=1를 표시한다).

따라서 현상태는 Q_2와 Q_1로 표시되고 각각 00, 01, 10, 11중 하나에 대응될 수 있다. 그리고 입력과 출력의 관계는 슬래시의 상단을 입력, 하단을 그 입력에 대한 출력으로 표시한다.

다음 상태로 넘어가는 천이시간을 T라고 했을 때 예를 들어 초기상태가 q_0이고 입력이 X(0)=0, X(T)=1, X(2T)=0, X(3T)=1, X(4T)=1로 차례로 들어온다면 상태변화는

$$q_0 \rightarrow q_2 \rightarrow q_3 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2$$

로 될 것이고 이에 해당하는 출력값은 Z(0)=0, Z(T)=1, Z(2T)=0, Z(3T)=0, Z(4T)=0가 된다.

그러나 초기 상태가 q_2일때는 동일한 입력변화에 대하여 상태변화가

$$q_2 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_2$$

가 되고, 이에 해당하는 출력값은 Z(0)=0, Z(T)=0, Z(2T)=0, Z(3T)=1, Z(4T)=0가 되어 초기상태가 다를 때 결과가 달라짐을 알 수 있다.

그림 14의 상태천이도는 표 1과 같이 상태표로 표시할 수 있다. 표 1(a)에서 q_0~q_3를 각각 Q_2Q_1상태의 00, 01, 10, 11로 대치하면 표 1(b)와 같다.

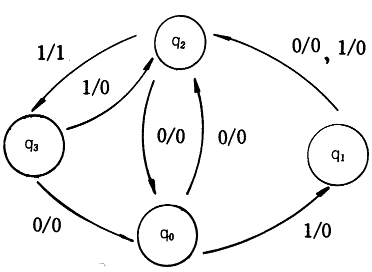
표 1(b)의 상태표로부터 기억회로로서 D 플립플롭을 사용하기로 하고 조합논리회로의 입력과 출력을 작성해보면 표 2와 같이 된다. 표 2로부터 D_1, D_2, Z는 모두 X, Q_2, Q_1의 함수로서 다음과 같이 되고 이들 관계를 만족하는 D_2, D_1, Z의 조합논리식을 구할 수 있으며 이를 구현하면 순서논리회로를 얻을 수 있다.

$$D_1 = f_1(X, Q_2, Q_1)$$
$$D_2 = f_2(X, Q_2, Q_1)$$
$$Z = f_3(X, Q_2, Q_1)$$

D_2, D_1 Z에 관한 논리식은 카르노맵을 작성하여 간략화된 식으로 표현할 수 있다. 표 3은 표 2의 내용을 카르노맵으로 나타낸 것이고, 표 3의 카르노맵으로부터 다음과 같은 D_2, D_1, Z에 대한 논리식을 얻게 된다.

$$D_2 = X'Q_2 + Q_2'Q_1 + XQ_2$$

〔그림 14〕 상태천이도의 예(x/z)



〔표 1〕 그림 14의 상태표

현상태 \ 현입력(X)	0 (다음상태/출력)	1 (다음상태/출력)
q_0	q_2/0	q_1/0
q_1	q_2/0	q_2/0
q_2	q_0/0	q_3/1
q_3	q_0/0	q_2/0

(a)

현상태(Q_2Q_1) \ 현입력(X)	0 (다음상태/출력)	1 (다음상태/출력)
00	10/0	01/0
01	10/0	10/0
10	00/0	11/1
11	00/0	10/0

(b)

$$D_1 = XQ_2'Q_1' + XQ_2Q_1'$$
$$Z = XQ_2Q_1'$$

그림 15는 표 3에서 얻은 논리식을 구현한 순서논리회로이다.

(2) 카운터

앞에서 언급한 순서논리회로를 이용해서 계수표에 따라 계수하는 카운터를 만들 수 있다. 카운터회로는 클럭신호

〔표 2〕 조합논리에 대한 순서논리회로의 입/출력

입력부			출력부		
현입력	현 상 태		D F/F의 입력		해당 출력
X	Q_2	Q_1	D_2	D_1	Z
0	0	0	1	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	0	0

〔표 3〕 D_2, D_1, Z의 카르노맵

Q_2Q_1 \ X	0	1
00	1	0
01	1	1
11	0	1
10	0	1

(a) D_2

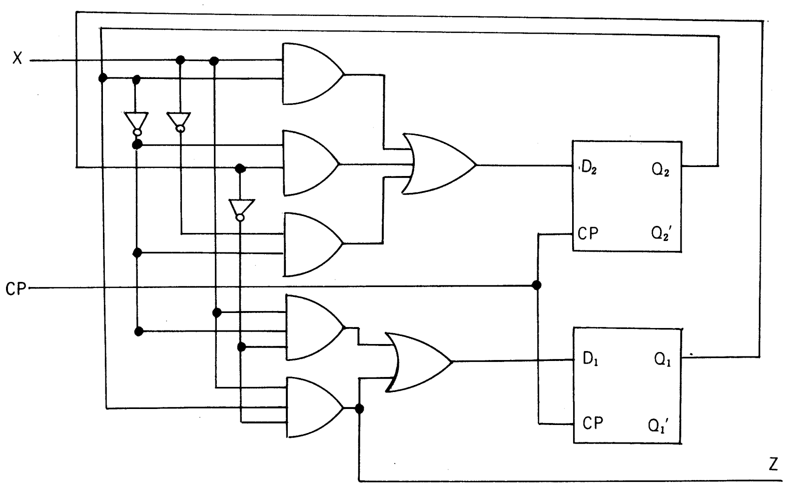
Q_2Q_1 \ X	0	1
00	0	1
01	0	0
11	0	0
10	0	1

(b) D_1

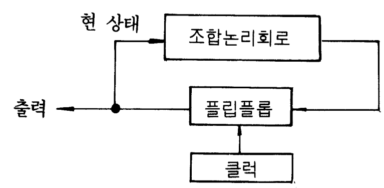
Q_2Q_1 \ X	0	1
00	0	0
01	0	0
11	0	0
10	0	1

(c) Z

〔그림 15〕 그림 14의 순서논리회로



〔그림 16〕 카운터의 블록도



가 인가되는 비율(클럭속도)에 따라서 동작하는 동기식 카운터회로와 클럭에 관계없이 동작하는 비동기식 카운터회로가 있다.

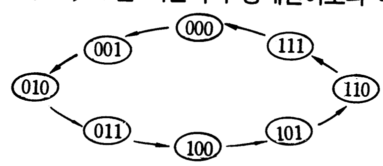
비동기식 카운터회로에서는 최하위 비트(LSB)를 나타내는 단(Stage)에만 클럭신호가 가해지고 나머지단에 대해서는 전단의 플립플롭의 현상태 Q 또는 Q′가 다음의 단 클럭 입력단자로 연결된다. 그러나 동기식 카운터 회로에서는 모든 플립플롭단이 하나의 클럭에 동기가 되어 동작한다.

카운터는 설계하는 방법에 따라 리플카운터(Ripple Counter), 궤환형카운터(Feedback Counter), 리세트형카운터, 직접리세트형 카운터 등의 비동기식 카운터와 모듈러스(Modulus) 동기식 카운터가 있다.

일반적으로 카운터는 정해진 진법의 수를 계수하게 되므로 임의의 코드가 일정한 순서를 갖고 순환하게 된다. 따라서 이렇게 일정한 순서를 갖고 순환하는 카운터회로의 경우에는 입력변수가 필요없고 출력변수 값은 기억회로의 현상태로 대응된다. 또한 다음상태로 현상태로부터 기대되므로 조합논리회로의 입력이 기억회로의 현상태가 되고 출력은 플립플롭의 입력이 된다. 여기서 기억회로는 플립플롭단을 의미한다.

이런 카운터의 블럭도는 그림 16과 같다.

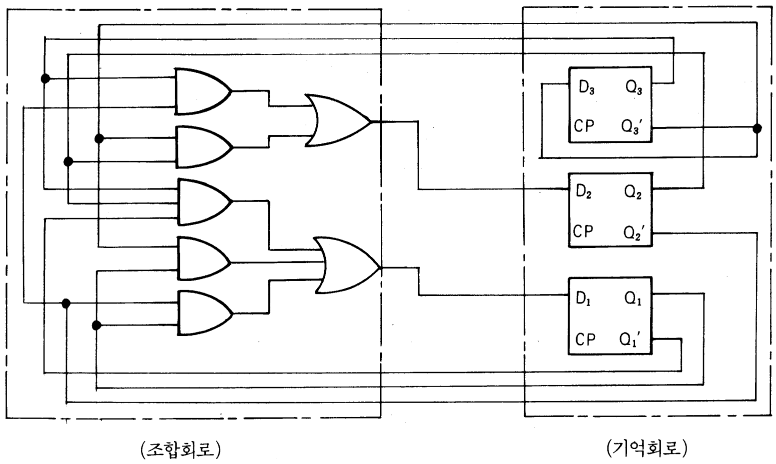
〔그림 17〕 8진 카운터의 상태천이도의 예



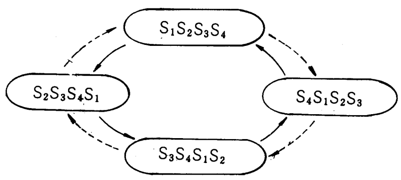
〔표 4〕 그림 17의 상태천이표

현 상 태			다 음 상 태		
Q_1	Q_2	Q_3	Q_1	Q_2	Q_3
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

〔그림 18〕 8진 카운터의 회로 구성



〔그림 19〕 시프트 카운터의 상태천이도
(각 S=0 또는 1)



〔표 6〕 표 5의 D_1, D_2, D_3에 대한 카르노맵

Q_2Q_3 \ Q_1	0	1
00	0	1
01	0	1
11	1	0
10	0	1

(a) D_1

Q_2Q_3 \ Q_1	0	1
00	0	0
01	1	1
11	0	0
10	1	1

(b) D_2

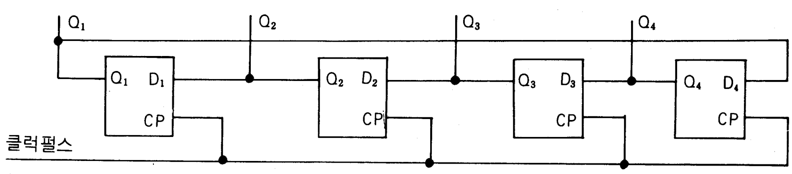
Q_2Q_3 \ Q_1	0	1
00	1	1
01	0	0
11	0	0
10	1	1

(c) D_3

〔표 5〕 표 4를 조합회로의 입력과 출력으로 표현

입력			출력		
현상태			D F/F의 입력		
Q_1	Q_2	Q_3	D_1	D_2	D_3
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

〔그림 20〕 시프트 카운터의 회로 구성도



예를 들어서 그림 17과 같은 8진 카운터의 상태천이도를 생각해 보자. 전체의 상태는 8개이므로 플립플롭은 최소한 3개(상태의 수$=2^n$, 여기서 n은 플립플롭의 수)가 필요하며 각각의 출력을 Q_1, Q_2, Q_3라고 하면 천이표는 표 4와 같이 된다.

기억회로를 D 플립플롭으로 한다면 조합논리회로의 입력단은 표 4의 현상태가 되고, 표 4의 다음상태는 D 플립플롭(기억회로)으로 기대되는 다음 상태가 된다.

다음은 D 플립플롭의 입력을 D_1, D_2, D_3로 하며 표 4를 표 5와 같이 다시 작성하고, 카르노맵에 표 6과 같이 적용하여 다음과 같은 간략화된 논리식을 얻는다. 그림 18은 이들 논리식을 구현한 것이다.

$$D_1 = f_1(Q_1, Q_2, Q_3) = Q_1'Q_2Q_3 + Q_1Q_3' + Q_1Q_2'$$
$$D_2 = f_2(Q_1, Q_2, Q_3) = Q_2'Q_3 + Q_2Q_3'$$
$$D_3 = f_3(Q_1, Q_2, Q_3) = Q_3'$$

이상과 같이 예를 들어서 카운터의 설계과정을 살펴 보았는데 부호순서가 특수한 경우에는 반드시 전술한 바와 같은 순서로 설계하지 않아도 된다. 예를 들어서 그림 19와 같이 동작되는 시프트 카운트(또는 링 카운터)를 생각해보면 그림 20과 같이 간단히 구성할 수 있다.

그림 19에서 실선방향으로 천이하는 경우 왼쪽시프트 카운터(Shift Left Counter)라고 하고, 점선방향으로 천이하는 경우는 오른쪽 시프트카운터(Shift Right Counter)라고 한다.

그림 20은 클럭펄스에 의해서 모든 플립플롭이 동기가 되므로 동기식 카운터이다. 디지탈회로에 많이 사용되고 있는 카운터는 IC화하여 시장에서 유통되고 있으며 이들에 클리어(Clear) 기능, 카운터의 Nonzero 초기값 설정 기능, 계수동작기능, 터미널계수 디코드기능 등과 같이 특수한 기능을 추가하기도 한다.



MS-DOS의 구조

마이크로소프트사에서 개발한 MS-DOS는 퍼스널컴퓨터용 DOS로서는 세계에서 가장 많이 보급되어 있다.

이러한 MS-DOS를 이해한다고 하는 것은 본격적인 OS라는 것을 이해하기 위한 제 1 보라고 말할 수 있을 것이다.

여기서는 평상시 우리들이 궁금하게 여기고 있는 화일시스템과 DOS의 내부구조 등을 소개하고 COMMAND COM과 MSDOS. SYS, IO.SYS 등이 서로 어떤 관계를 유지하며, 코맨드의 입력과 실행 등은 어떻게 이루어지는 지를 설명한다.

●특집●



MS-DOS의 구조

마이크로소프트사의 MS-DOS는 퍼스널컴퓨터용 DOS(Disk Operating System)로서는 세계에서 가장 많이 보급되어 있다는 것은 누구나가 잘 알고 있는 사실이다.

그림 1에 MS-DOS의 지금까지의 지내온 과정을 나타낸다. 그림에서 보면 알 수 있듯이 MS-DOS의 모체로 된 것은 시애틀·컴퓨터프로덕트사의 『86-DOS』이며 이것이 나중에 MS-DOS 버전 1.10으로서 세상에 등장하였다. 이 버전을 탑재한 최초의 머신은 IBM PC이며, 이후 PC-DOS라고 불려졌다.

1981년에 IBM PC(CPU는 i8088)용으로서 릴리즈하였을 때는 8비트용 DOS로서 CP/M이 전성기였던 배경도 있겠지만, 아뭏튼 최초로 등장한 버전1.10은 CP/M과 거의 동등한 기능과 구조를 가진 DOS였다. 이것이 1983년에 릴리즈된 버전2.0에서는 당시 주목되었던 UNIX의 기능중 장점만을 도입, CP/M과 유사한 DOS에서 탈피하여 UNIX와 유사한 DOS로서의 성격이 강하게 표시되었다.

지금은 퍼스컴상에서 UNIX를 동작시켜 UNIX 오퍼레이션을 일반의 퍼스컴유저가 피부로 직접느끼고 이해할 수 있도록 되었지만 당시에는 UNIX자체가 일부의 기업이나 개발자, 대학 등 일부층에서만 사용할 수 있어서 일반 퍼스컴유저로서는 감히 접근조차할 수 없던 시기였다.

계층구조디렉토리, 파이프나 리다이렉션 등은 MS-DOS의 기능중 하나로서 당연한 것으로 되어 있지만 MS-DOS가 CP/M과의 경쟁에서 이길 수 있었던 것은 UNIX의 기능을 도입한 것으로 이것이 하나의 요인으로 작용하였던 것이다.

이것은 MS-DOS를 개발한 마이크로소프트사의 선견지명이라고 말할 수 있다.

단지 컴퓨터를 이용하기만 한다면 MS-DOS의 내부구조를 알 필요가 없겠지만 개발한다거나 할때는 반드시 그 내부구조를 이해해야 한다는 것은 자명한 사실이다. 또 MS-DOS에 대해서 배우는 것은 본격적인 OS라는 것을 이해하기 위한 제1보라고 말할 수 있을 것이다.

그림 2는 MS-DOS의 기본적인 구조를 나타낸 것이다.

화일시스템

OS의 중요한 일중의 하나로 화일관리라는 것이 있다. 간단히 말하면 디렉토리를 만들어 각각의 화일을 유저에게 보이는 형(보이지 않을 경우도 있지만)으로 관리하는 것이다. 이 관리방법은 OS의 성격을 형성하는 요인의 하나이며, 유저가 OS가 하는 일로서 가장 인식하기 쉬운 것일 것이다.

MS-DOS의 화일관리 수법을 보면 초기의 버전에서는 CP/M과 거의 동등한 단순한 것이였다. 이것도 버전2.10 이후에서는 UNIX의 영향을 받아서 계층구조 디렉토리를 취급하도록 되었다. 다음에 이러한 화일관리의 구조를 알아보자.

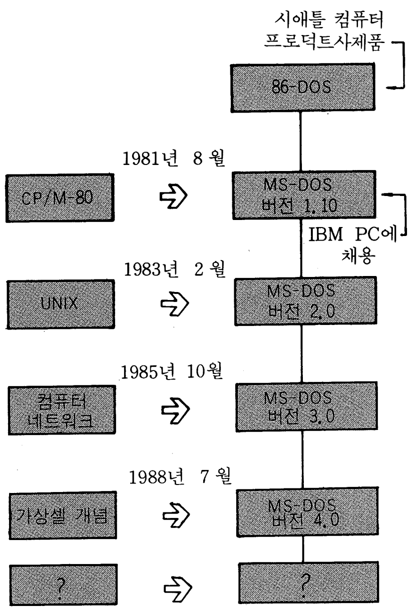
(1) 디렉토리엔트리

MS-DOS의 표준 플로피디스크에서는 디스크상의 영역을 **그림 3**과 같이 구별하고 있다. 이들은 외측부터 시스템영역, FAT(File Allocation Table)영역, (루트)디렉토리영역, 데이터영역(또는 클러스터영역)이라고 불리우고 있다.

시스템예약영역에는 MS-DOS를 부트하기 위한 부트프로그램이 수록되어 있다. FAT영역과 디렉토리영역은 이제부터 설명하는 화일관리의 구조에 크게 관계되고 있는 부분이다. 그리고 나머지 데이터영역은 말 그대로 데이터, 즉 화일을 기록하는 부분이다.

유저가 DIR코맨드를 실행하였을 때에 표시되는 화일명이나 사이즈, 날짜 등은 통상은 디렉토리영역에 수록되어 있다. 이 디렉토리영역중에 있는 화일의 정보는 디렉토리엔트리라고 불리우며 **그림 4**와 같은 구조를 하고 있다. DIR코맨드로 표시되는 내용은 이 속의 화일명, 확장자(화일타입), 화일갱신시각, 갱신년월일, 그리고 화일의 사이즈이다. 이것 이외에는 시스템예약영역으로서 현재는 사용되고 있지 않는 부분으로 화일속성과 화일이 수록되어 있는 선두의 클러스터번호를 기록하고 있는 부분이 있다.

〔그림 1〕 MS-DOS의 발전과정



여기서 화일속성에 대해서 알아보자. 현 버전(3.10이상)에서는 이 속성값은 **표 1**과 같이 정해지고 있다. 예를 들면 이 속성값이 02H의 화일은 『숨겨진 화일(Hidden File)』이라고 불리우고 있으며, DIR코맨드에 의해서 표시시킬 수 없을 뿐만 아니라 DEL코맨드로도 삭제할 수 없다. 이런 형태의 예로서는 MS-DOS의 시스템화일인 『MSDOS.SYS』와 『IO.SYS』가 있다. 또 이 값이 10H인 화일은 통상의 화일은 아니고 서브디렉토리인 것을 나타내고 있다. 이 서브디렉토리에 대해서는 나중에 설명하기로 한다.

버전2.xx까지의 MS-DOS에서는 화일속성을 유저가 변경하는 코맨드는 마련되어 있지 않았다. 버전3.10이상에서는 ATTIRB코맨드에 의해서 리드전용/라이트금지속성을 변경할 수 있도록 되어 있다.

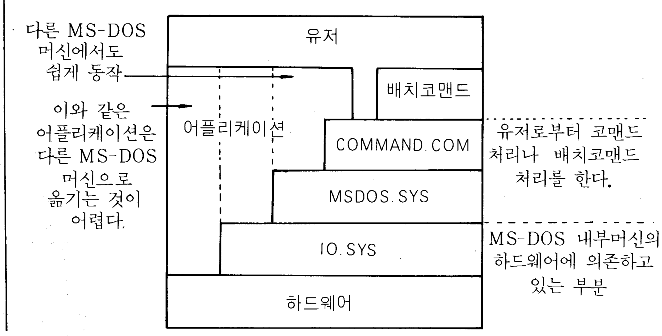
(2) FAT

MS-DOS 특징의 하나로서 가지각색인 크기의 디스크를 적어도 유저 입장에서 보면 아무 것도 인식하지 않고 동등하게 취급하는 점을 들 수 있다. 더욱 다른 형태의 디스크간에도 비교적 높은 호환성을 유지하고 있는 것도 우수한 점이라고 할 수 있다. 이것을 실현하기 위한 메카니즘에 대해서는 나중에 상세하게 설명하기로 하고 여기서는 이들을 어떻게 하여 식별하고 있는가에 대해서 알아보자.

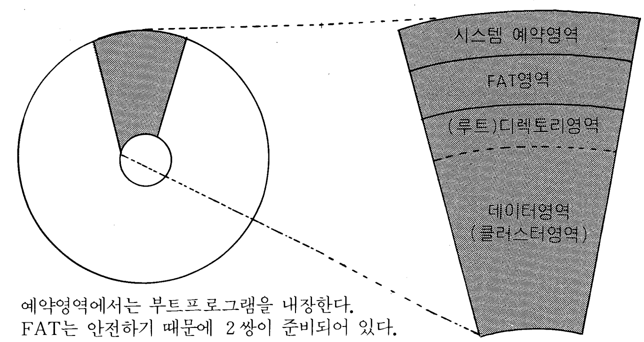
현재 3.5인치, 5.25인치, 8인치의 각 플로피디스크에서 MS-DOS의 표준포맷이 정해져 있다. 이것은 **표 2**와 같이 되어 있다. 이 표에서 보면 FAT ID라고 불리우는 것이 있는데 이것은 그 미디어의 물리적인 타입을 나타내는 것이다. 그리고 이 FAT ID는 앞에서 설명한 디스크영역의 FAT영역 선두에 수록되어 있다. 즉 MS-DOS는 이 FAT ID를 참조함으로써 지금 액세스하고자 하는 디스크의 타입을 식별하는 것이다. 예를 들면 같은 FAT ID를 가지는 3.5인치 2HD와 5.25인치 2HD의 플로피디스크는 MS-DOS의 시스템에서 보면 전부 동등하게 취급되는 것을 알 수 있다.

한편 FAT본체는 디스크의 사용상황을 기록하는 테이블이다. 즉 FAT에는 화일의 위치, 그 화일이 사용하고 있는 섹터, 빈 섹터 등의 정보를 기록하고 있다. FAT는 클러스터라고 불리우는 것을 1단위로서 디스크의 데이터영역을 관리하고 있다. 예를 들면 5.25인치 2DD에서는 1클러스터당의 섹터수는 2로 되어 있다. 또 이 표에서 보면 1섹터는 512바이트인 것을 알 수

〔그림 2〕 MS-DOS의 구조



〔그림 3〕 디스크상의 영역



있다. 즉 5.25인치 2DD의 1클러스터는 1024바이트의 용량을 가지는 것이다. 다른 미디어타입에 대해서도 1클러스터는 1024바이트(8인치 1S만은 512바이트)이다.

클러스터는 MS-DOS가 취급하는 용량의 최소단위이므로 어떤 짧은 화일에서도 최저 1024바이트는 디스크의 데이터영역으로 사용되어야 한다는 것을 알 수 있다. 또 DIR코맨드로 표시되는 사용바이트수는 실제로 유효한 데이터가 격납되어 있는 부분만의 크기이고, 그 화일이 그 디스크상에서 점유하고 있는 영역은 아니라는 것이다.

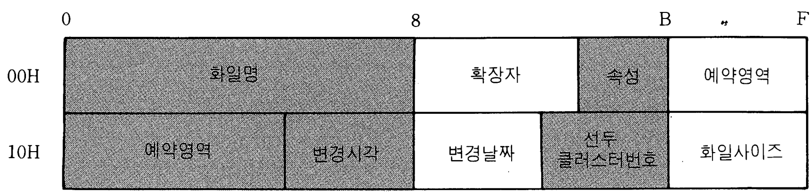
〔표 1〕 화일 속성치

속성치	의 미
01H	읽어내기 전용화일
02H	숨겨진 화일
04H	시스템 화일
08H	볼륨 레이블
10H	디렉토리
20H	보존 화일

즉 어플리케이션을 기동하기 위한 배치화일(배치화일에 대해서는 나중에 설명한다)을 디스크상에 많이 만들어 두면 그 하나 하나의 화일 크기가 가령 100바이트 밖에 사용하고 있지 않더라도 디스크상의 영역에는 최저 1024바이트가 사용되어 버린다고 하는 것이다. 그러므로 유저가 DIR코맨드에서 화일의 용량을 표시하기 보다도 실제의 디스크상의 사용영역은 많아진다. 또 하드디스크를 사용하고 있을 경우는 1클러스터의 사이즈가 8192바이트이므로 1K나 2K바이트정도의 작은 화일이 많이 수록될 경우는 플로피시스템보다도 더욱 디스크의 사용효율이 나빠져 버린다.

하드디스크의 용량은 증가하는 한편 크기는 점차 소형화되어 가고 있는 추세이다. 그러나 현재의 MS-DOS에서는 하드디스크 용량의 대소에 관계없이 1클러스터의 용량은 고정되어 있다. 앞으로의 하드디스크 용량의 증대를 생각하면 용량에 따라서 MS-DOS시스

〔그림 4〕 디렉토리 엔트리



〔표 2〕 대표적인 디스크 코맨드

명 칭	3.5″ 2DD	3.5″ 2DD	3.5″ 2HD	5.25″ 2DD	5.25″ 2DD	5.25″ 2HD	8″1S	8″1S	8″2D
트랙수	80	80	77	80	80	77	77	77	77
섹터/트랙	7	8	8	9	8	8	26	26	8
면수	2	2	2	2	2	2	1	1	2
섹터길이	512	512	1024	512	512	1024	128	128	1024
디스크용량	720	640	1230	720	640	1230	250	250	1230
루트디렉토리(MAX)	112	112	192	112	112	192	68	68	192
섹터/클러스터	2	2	1	2	2	1	4	4	1
예약 섹터수	1	1	1	1	1	1	1	4	1
FAT수	2	2	2	2	2	2	2	2	2
섹터/FAT	3	2	2	3	2	2	6	6	2
FAT ID	F9H	FBH	FEH	F9H	FBH	FEH	FEH	FDH	FEH

템이 최적인 1클러스터용량을 설정하여 액세스효율과 사용효율을 증가시키는 방법이 필요할 것이다.

DOS와 취급하는 화일액세스의 최저용량은 디스크와 메모리간의 전송효율에 크게 영향을 주기 때문에 DOS의 설계자에게 가장 골치아픈 부분이다. 디스크의 사용효율을 높인다면 1클러스터를 1섹터와 같게 하면 되는 것이지만 디스크의 리드/라이트나 메모리와의 전송속도는 저하되어 버린다. 반대로 1클러스터의 용량을 크게 하면 디스크의 리드/라이트시간 및 메모리와의 전송시간을 단축할 수 있지만 디스크의 사용효율이 저하되어 버린다. 또 디스크의 액세스속도를 결정짓는 요인은 1클러스터의 용량이나 버퍼사이즈의 설정만이 아니고 스큐도 밀접하게 관계하고 있다.

스큐도 몇 섹터마다 리드/라이트를 하는가를 설정하는 것이며, 디스크의 회전속도나 DOS자체의 액세스속도, 버퍼사이즈, 1클러스터의 용량과의 관계로 정해진다. CP/M-80(8비트 CPU용)에서는 소스가 유저에게 공개되어 있었으므로 사용하고 있는 디스크에 맞추어 최적인 스큐를 설정할 수 있었지만 MS-DOS에서는 미디어타입에 의해 고정되어 있다. 스큐의 개념을 그림 5에 나타낸다.

그러면 여기서 다시 그림 4의 디렉토리엔트리를 생각해 보자. 이 속에는 그 화일의 선두 클러스터번호라는 것이 있었다. 그러나 이것만으로는 그 화일이 디스크상의 데이터영역 어느 부분을 사용하고 있는 것인지 알 수 없다. 이때 이것을 나타내고 있는 것이 FAT인 것이다. FAT의 하나의 요소(FAT엔트리)는 12비트의 크기를 가지고 있다. 실제로는 2개의 FAT엔트리에서 3바이트를 차지하고 있는 것이다. 이 1개의 FAT엔트리는 데이터영역상의 하나의 클러스터에 대응하고 있다. 그리고 빈 클러스터에 대해서는 000H가, 불량클러스터에 대해서는 FF7H

라고 하는 특별한 값이 라이트되고 있다.

가장 중요한 것은 사용중인 클러스터인데 이것에는 그 클러스터를 사용하고 있는 화일에 할당되어 있는 다음 클러스터의 번호가 라이트된다. 즉 어떤 화일이 사용하고 있는 클러스터를 알려면 우선 디렉토리엔트리에 의해서 선두 클러스터번호를 조사한다. 다음에 그 클러스터번호에 대응하는 FAT 엔트리에 쓰여진 클러스터번호를 보면 2번째 클러스터를 알 수 있다. 이것을 반복하여 화일의 최종클러스터인 것을 나타내는 FFFH가 FAT엔트리에 쓰여져 있는 것까지 더듬어 가면 화일에 할당된 모든 클러스터를 순서대로 알 수 있는 것이다. 이 모양을 그림6에 나타낸다.

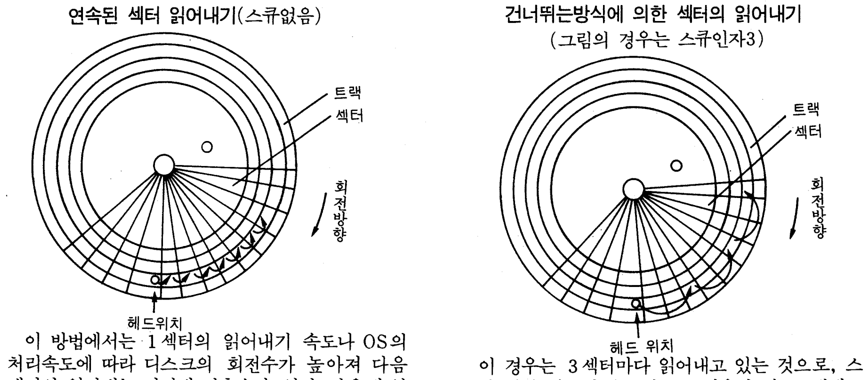
(3) 계층디렉토리

디스크의 대용량화에 수반하여 1개의 디스크에 수록할 수 있는 화일의 수도 자연히 증가한다. 이렇게 되면 평탄한 구조의 디렉토리에서는 목적 화일을 찾는데에 힘이 들고, 화일의 관리상태도 마음에 드는 것은 아니다. 그래서 등장하는 것이 계층구조디렉토리라고 하는 것이다. 그림 7은 계층화디렉토리의 예를 나타낸 것이다. 화일의 종류나 유저별로 디렉토리를 설정하여 두면 화일의 검색을 효율좋게 할 수 있다. 또 잘못해서 다른 유저의 화일을 삭제하여 버리도록 하는 위험도 회피할 수 있다.

MS-DOS가 재빨리 이 개념을 도입한 것은 CP/M과 비교해서 절대적 우위를 확립한 요인의 하나로 생각할 수 있다. 여기서는 이 계층화디렉토리를 실현하는 구조의 문제점을 알아본다.

디렉토리엔트리 속의 화일속성의 하나로 서브디렉토리라고 하는 것이 있었다. 이 서브디렉토리라는 사고방식이 계층구조를 가지는 디렉토리를 실현하기 위한 기본적 개념으로 되어

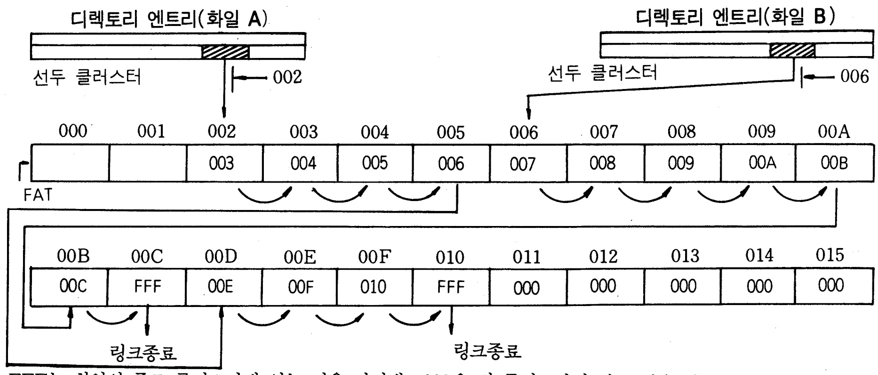
〔그림 5〕 스큐의 개념



이 방법에서는 1섹터의 읽어내기 속도나 OS의 처리속도에 따라 디스크의 회전수가 높아져 다음 섹터의 읽어내는 간격에 맞출수가 없어 다음에 읽을만한 섹터가 1회전할 때까지 기다려야 하기 때문에 액세스효과가 저하된다.

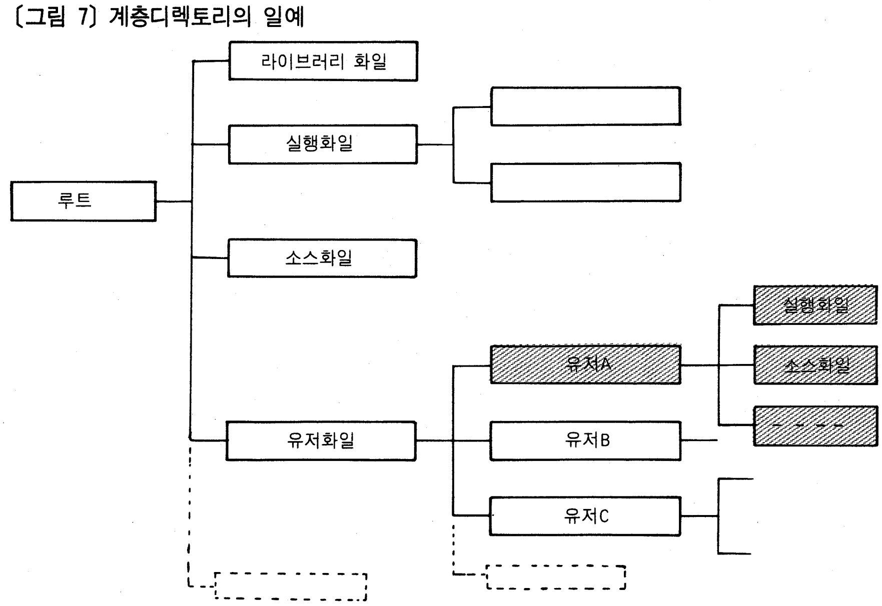
이 경우는 3섹터마다 읽어내고 있는 것으로, 스큐가 없기 때문에 속도적으로 여유가 있고, 액세스 효율의 저하 및 리드에러의 발생을 억제하는 일이 생긴다.

〔그림 6〕 FAT 링크



FFF는 화일의 종료 클러스터에 있는 것을 나타냄. 000은 빈 클러스터에 있는 것을 나타냄

〔그림 7〕 계층디렉토리의 일예



헤아릴수 있는 디렉토리중에 사선으로 표시된 유저 A에 속해 있는 것은 한눈에 볼 수 있다.

있다. 즉 우선 루트디렉토리의 속에 몇개인가의 서브디렉토리가 화일로서 등록되어 있다. 그리고 각각의 서브디렉토리의 가운데에는 더욱 또 몇개인가의 서브디렉토리가 역시 화일로서 등록되어 있다. 이것은 그 디스크의 데이터영역의 용량이 허락하는 한 몇겹이라도 계속할 수 있다. 이 모양을 그림으로 나타내면 **그림 8**과 같이 된다.

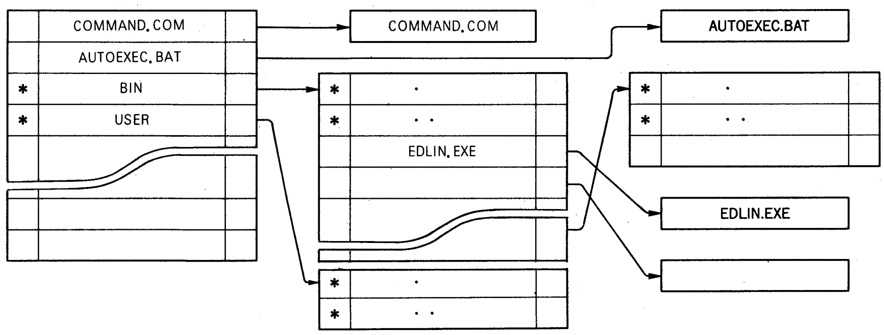
서브디렉토리에서의 화일의 내용은 디렉토리의 총수와 같고 루트디렉토리의 등록화일수의 제한과는 달라서 서브디렉토리에는 화일로서의 용량제한(논리적으로는 FFFFFFFFH바이트)밖에 없으므로 실제로는 그 디스크의 용량이 허락하는 한 몇개라도 화일을 등록하는 것이 가능하다. 주의해야 할 것은 FAT에는 루트디렉토리와 서브디렉토리용의 것이 따로 존재한다고 하는 것 같은 것은 없다. FAT는 디스크의 클러스터에 대응하는 것이며 특정한 화일에 속하고 있는 것은 아니기 때문이다. FAT는 화일측에서 특정한 FAT엔트리의 다음 위치를 지시하고 있는데에 지나지 않는다. 그 화일이 루트디렉토리의 것인지, 서브디렉토리의 것인지 전혀 구별되지 않는다.

그런데 서브디렉토리에는 루트디렉토리에는 존재하지 않는 2개의 화일(속성은 서브디렉토리)이 있다. 이 2개의 화일명은 "."과 ".."이다. 이들은 각각 그 디렉토리자신(즉 자기자신의 디렉토리를 나타낸다)과 그 디렉토리의 모디렉토리를 나타낸다.

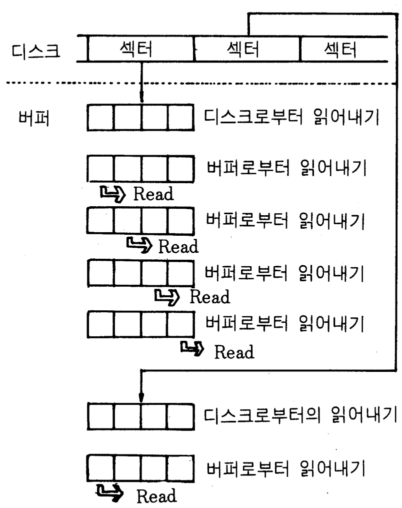
이들의 화일에 대한 디렉토리엔트리의 내부구조를 알아보자. 그 화일의 선두클러스터를 나타내는 필드에는 각각 자신이 디렉토리 및 모디렉토리가 수록되어 있는 선두클러스터번호가 라이트되어 있다.

도대체 어떻게 해서 이런 것이 필요한 것인가? 계층상으로 뒤얽힌 디렉토리의 속을 이동하여 여기저기에 있는 화일을 액세스하는 경우를 생각해 보자. 이럴 때 만일 자기가 MS-DOS였다면 지금 자기는 어디에 있는 것인가? 자기는 어디로 돌아가면 좋을 것인가? 등이 불안하게 되는 것이다. 자기자신이 어디에 있는가를 지시하는 표식이 없어져 버리면 MS-DOS의 계층화디렉토리는 기호 이외의 아무 것도 아닌 것이다.

〔그림 8〕 계층디렉토리의 구조



〔그림 9〕 디스크의 버퍼링크



그러나 계층화디렉토리가 좋은 면만을 가지고 있다고는 생각되지 않는다. 계층화디렉토리가 서포트되어 있는점과는 달리 디렉토리를 조작하는 명령이 빈약한 것도 사실이다. 표준으로 서포트되어 있는 것이 MKDIR(디렉토리의 작성), RMDIR(디렉토리의 삭제), CHDIR (디렉토리의 이동)의 3종류뿐이며, 디렉토리내의 화일을 포함한 디렉토리의 이동이나 디렉토리내에 화일이 있는 상태에서의 디렉토리 삭제 등의 명령은 서포트되어 있지 않으므로 디렉토리의 정리 등을 하는 경우 작업에 시간이 많이 걸리게 된다. 플로피디스크의 경우는 1M바이트정도의 용량이므로 이것은 그다지 큰 문제로는 되지 않지만 10M바이트이상의 하드디스크를 사용하는 경우는 이들은 큰 문제점으로 등장한다.

계층화디렉토리를 사용할 수 있기 때문이라고 해서 함부로 디렉토리를 작성하는 것은 나중에 정리하는 경우에 귀찮게 될 뿐더러 디렉토리자체도 디스크의 용량을 점점 먹어간다고 하는 것을 충분히 생각해 두지 않으면 안된다.

(4) 버퍼링

어플리케이션이나 유저의 프로그램이 화일을 액세스하는 경우 한번에 리드/라이트하는 바이트수는 가지각색이다. 한편 하드웨어에 의한 액세스의 단위는 디스크에 정해진대로 1섹터단위로 결정되어 있다. 이 차이를 잘 흡수하기 위해서 버퍼링이라고 불리우는 수법이 사용되고 있다. 예를 들면 화일을 리드할 때에는 우선 한번에 1섹터의 내용이 디스크에서 메모리속의 버퍼로 전송된다. 그 후 프로그램의 요구에 따른 바이트수가 순번대로 이번에는 이 버퍼에서 꺼내지는 것이다. 그리고 버퍼가 비게 된 시점에서 다음의 섹터의 내용이 버퍼에 전송되며 이어

서 이 버퍼에서 필요한 바이트수가 꺼내진다고 하는 방식이다. 이 모양을 **그림 9**에 나타낸다. 라이트시에는 이것과 꼭 반대의 형태로 이루어지는 것이다.

디렉토리나 FAT는 빈번하게 액세스되기 때문에 이것에 대해서 버퍼링을 효율좋게 하려면 액세스하는 경우에는 프로그램에서 오픈이나 클로즈의 요구가 있을 때에 실제의 섹터내용과 버퍼의 내용을 일치시키면 되는 것이다. 그렇지만 디렉토리의 액세스에는 오픈도 클로즈도 없다. 그때문에 이들에 대해서 버퍼링을 효율좋게 하기 위해서는 디스크가 교환된 것을 검출할 수 있는 하드웨어가 필요하게 된다.

버퍼링에 사용되는 버퍼의 수는 유저가 자기 머신의 메모리용량이나 용도에 따라서 자유로이 설정할 수 있다. 예를 들면『CONFIG.SYS』화일에

```
BUFFERS=10
```

라 쓰여 있으면 이 화일을 포함하는 디스크를 부트하면 자동적으로 10개의 버퍼가 확보된다. 설정할 수 있는 버퍼의 최대수는 99로 되어 있지만 함부로 많이 하는 것은 생각해 보아야 할 문제이다. 이러한 것은 그 분량만큼 확실히 메모리가 소비되고 버퍼를 취급하는 처리의 오버헤드 등에서 오히려 시스템 전체의 효율이 떨어지지 않는다고 말할 수 없다.

버퍼 1개당의 메모리용량은 약1K바이트 필요하므로 만일 99개의 버퍼를 확보하였다고 하면 실로 100K바이트에 가까운 용량의 메모리가 버퍼로 잡혀 버리게 된다. **리스트 1**은 CHKDSK 코맨드를 이용하여 이러한 내용을 나타낸 것이다. 설정하고 있는 버퍼의 수와 유저가 사용가능한 메모리 용량의 표시에 주목하기 바란다.

워드프로세서나 에디터를 사용하는 경우는 버퍼를 많게 하면 버퍼내에 수록할 수 있는 화일용량이 많아지기 때문에 큰 화일을 에디트하여도 디스크에 대한 액세스횟수는 매우 작아진다. 그러나 에디트를 하는 메인메모리는 확실히 적어지고, 그만큼 처리효율은 확실히 저하하여 버린다. 최근의 워드프로세서나 에디터에서는 범용의 RAM 디스크를 이용하여 처리속도와 효율을 높이고 있다.

〔리스트 1〕 버퍼의 확보

```
A>CHKDSK

   362496 bytes total disk space  ;총 디스크용량
    41984 bytes in 2 hidden files ;2개의 시스템화일
   307200 bytes in 29 user files  ;29개의 유저화일
    13312 bytes available on disk ;사용가능한 디스크용량

   655360 bytes total memory      ;총 메모리
   599600 bytes free              ;사용가능한 메모리
A>


A>TYPE CONFIG.SYS
BREAK = ON ;프로그램 실행시 Ctrl+Break 상태를 검사.
           디폴트 값은 OFF로 되어 있다.
BUFFERS = 10 ;메모리상의 디스크 버퍼수 설정.
             1에서 99까지 지정가능하고 디폴트값은 XT는 2, AT는 3으로
             되어 있다.
SHELL = name ;SHELL로 지정한 프로그램을 실행.
             디폴트값은 SHELL=COMMAND.COM으로 되어 있다.

A>
```

실장메모리용량에도 비례하지만 640K바이트 이상의 메모리가 실장되어 있을 경우에도 20~30개 정도의 버퍼를 확보하여 두면 대부분의 어플리케이션에 있어서 디스크액세스가 적은 쾌적한 환경이 얻어진다. 또 유저가 버퍼를 설정을 하지 않을 경우는 MS-DOS시스템기동시의 디폴트버퍼로서 2개(버퍼사이즈 약2K바이트)를 확보한다.

내부구조

일반적인 OS의 일로서는 화일관리, 이외에 잡관리, 프로세스관리, 메모리관리, 입출력처리 등을 들 수 있다. MS-DOS는 이들의 처리를『COMMAND.COM』,『MSDOS.SYS』,『IO.SYS』라고 불리우는 3개의 화일에 의해 실현하고 있다. 이들은 계층적으로 결합하여 어플리케이션이나 유저에서의 요구를 스마트하게 처리하고 있다. 여기서는 각각의 기능과 구조를 알아보자.

(1) COMMAND.COM

COMMAND.COM은 MS-DOS의 코맨드프로세서의 역할을 다하고 있다. 즉 유저가 키보드 등에서 입력한 코맨드(문자열)를 해석하여 그것에 대응하는 처리를 한다. 여기에서도 UNIX의 코맨드프로세서인 셸의 개념이 도입되어 I/O리다이렉션이나 파이프의 기능이 실현되고 있다. 그렇다고 하여도 COMMAND.COM은 그들을 실행하기 위한 셋업을 할 뿐이며, 실제로는 MSDOS. SYS의 기능에 의해서 실행되는 것이다. 여기서는 COMMAND.COM의 기능과 그것을 실현하기 위한

구조에 대해서 설명한다.

●코맨드처리

MS-DOS의 일반적인 코맨드 형식은

코맨드형 파라미터 파라미터

와 같은 형을 하고 있다. 이 가운데 COMMAND. COM이 해석하는 것은 코맨드명의 부분이며, 파라미터는 기동된 코맨드(프로그램)에 주어진다.

내부코맨드란 COMMAND.COM자신이 그 처리방법을 알고 있는 코맨드이다. 즉 COMMAND.COM자신의 속에 그것을 실행하는 프로그램이 들어있는 것이다. 사용빈도가 특히 높다고 생각되는 코맨드는 거의 이 내부코맨드로서 마련되어 있다. DIR, DEL, RENAME, TYPE나 CHDIR, MKDIR, RMDIR 등이 그 대표적인 것이다.

한편, 외부코맨드란 그 처리방법이 COMMAND.COM의 외부에 화일로서 마련되어 있는 것이다. 외부코맨드는 3종류로 분류되어 이들은 화일의 확장자명으로 구별할 수 있다. 즉 화일명의 뒤에 "COM", "EXE" 그리고 "BAT"과 같이 3종류로 나눠진다. 이 3문자는 확장자라고 불리우며, 화일의 타입이나 실행형식 등에 따라 나누어 사용된다. 이가운데 앞의 2개는 MS-DOS상에서 CPU가 직접 실행가능한 기계어 프로그램의 화일이다.

이들의 차이를 간단하게 설명하면 프로그램과 데이터를 합쳐서 64K바이트 이내에 수록되는 것은 "COM", 그 이상의 크기로 되어 버리는 것은 "EXE"라고 하는 확장자가 사용된다.

COM형식의 화일은 원래 CP/M-80과의 호환성을 생각하여 채용된 것이지만 지금은 그 흔적이라 생각하는 쪽이 적당할 것이다. 단, 8086/88CPU에서는 세그먼트레지스터를 변경하지 않고 직접액세스할 수 있는 영역이 64K바이트로 제한되어 있기 때문에 COM형식의 화일쪽이 효율이 좋은 것은 사실이다.

〔표 3〕 표준 입출력 화일핸들

번 호	명 칭	디바이스명	표준상태에서의 기능
0	표준 입력	CON	콘솔에서 입력
1	표준 출력	CON	콘솔로 출력
2	표준 에러 출력	CON	항상 콘솔로 출력
3	표준 보조장치	AUX	RS-232C 입출력
4	표준 리스트 표준	PRN	프린터로 출력

"BAT"의 확장자명을 가지는 화일은 배치화일이라고 불리우고 있다. 이것은 COMMAND.COM에 주는 코맨드를 문자열로서 나열한 것 같은 형을 한 텍스트화일이다. 이것에 대해서는 나중에 상세하게 설명한다.

이 3개의 확장자 이외의 확장자도 사용할 수 있다. 단, COM, EXE, BAT가 MS-DOS시스템의 예약확장자로 되어 있는 것 처럼 이것 이외의 확장자도 다른 언어 등에서 사용되고 있기 때문에 유저가 잘 이해하여 확장자를 붙이는 것이 나중에 혼란을 초래할 가능성을 없앨 수 있다. C언어의 소스는 "C", 어셈블러소스는 "ASM" 포트란의 소스는 "FOR"라고 확장자명을 붙이며, 이런 것들은 이미 예약된 확장자이다. 또 "SYS"는 일반으로 시스템화일 또는 디바이스드라이버의 확장자로서 사용되고 있다.

COMMAND.COM은 코맨드가 주어지면 우선 그것이 내부코맨드인지 어떤지를 조사한다. 내부코맨드면 즉석에 실행되고, 외부코맨드의 경우는 디스크내의 화일이 검색된다. 그 실행우선순서는 "COM", "EXE", "BAT"의 확장자명을 가지는 각 화일의 순으로 되어 있다. 이러한 것은 예를 들면 같은 이름을 가지는 "COM"형식의 코맨드와 "BAT"형식의 코맨드가 있었을 경우 "COM"형식의 것 만이 실행되고, "BAT"형식의 것은 실행되지 않는다.

"BAT"형식의 코맨드화일이 실행될 때에는 그 화일의 내용이 마치 키보드에서 타이프된 것과 같이 처리된다. 이것을 실현하고 있는 MS-DOS의 중요한 메카니즘이 이제부터 해설하는 I/O리다이렉션기능이다.

●I/O리다이렉션

I/O리다이렉션도 UNIX에서 받아들인 중요한 개념의 하나이다. 이 기능의 기본은 "표준입출력"이라는 사고방식에 의거하여 입출력을 하는 것이다. 이것에 의해서 코맨드나 어플리케이션에 대한 입력선이나 그들의 처리결과의 출력선을 자유로 전환할 수 있다.

이 사고방식의 포인트는 입출력기기를 모두 화일로서 취급하는 것에 있다. MS-DOS는 화일을 액세스할 때 화일마다 화일핸들이라는 것을 마련한다. 한번 화일이 오픈된 후는 화일명은 아니고 이화일핸들에 대해서 리드·라이트의 지시가 이루어진다. 표준적인 입출력기기에 대해서는 **표 3**과 같이 0번에서 4번까지의 화일핸들이 미리 마련되어 있다. 이 화일핸들을 교체함으로써 입출력선을 자유자재로 전환하도록 하는 기능이 『I /O리다이렉션』의 사고방식이다.

코맨드라인에 "〈"나 "〉" 혹은 "〉〉"의 문자가 포함되어 있으면 COMMAND. COM은 입력 혹은 출력선을 표준입출력인 콘솔(키보드와 CRT)에서 지정된 화일로 전환한다. 여기서 화일(즉 입출력선)의 지정은 표 3에 있는 디바이스명이나 실제의 디스크상에 존재하는(또는 작성하는) 화일명에서 한다. 예를 들면 **리스트 2**의 예와 같이 DIR코맨드를 리다이렉션을 사용하여 FILE1로 보냈다고 하면 드라이브A의 디렉토리 내용은 "FILE1"이라는 화일에 라이트된다.

이것은 우선 FILE1이 라이트모드에

서 오픈되고 이 화일핸들이 표준출력(통상은 CRT)의 화일핸들인 1번에 할당된다. 이 모양을 그림 10에 나타낸다. 그후 DIR코맨드가 실행되어 결과

〔리스트 2〕 I/O 리다이렉션

```
A:\>dir

 Volume in drive A has no label
 Directory of  A:\

ANSI     SYS      1678   3-17-87  12:00p
APPEND   EXE      5825   3-17-87  12:00p
:
P1                 768   3-07-89   2:22p
FILE1             1368   3-07-89   2:23p
        31 File(s)      10240 bytes free

A:\>dir >file1

A:\>type file1

 Volume in drive A has no label
 Directory of  A:\

ANSI     SYS      1678   3-17-87  12:00p
APPEND   EXE      5825   3-17-87  12:00p
:
P1                 768   3-07-89   2:22p
FILE1                0   3-07-89   2:25p
        31 File(s)      10240 bytes free
```

〔리스트 3〕 리다이렉션의 어펜드모드

```
A>TYPE PARK
나는 초보자입니다.

A>TYPE KIM
월간 퍼스널컴퓨터는
초보자를 위한 컴퓨터전문지입니다.

A>TYPE PARK >> KIM

A>TYPE KIM
월간 퍼스널컴퓨터는
초보자를 위한 컴퓨터전문지입니다.
나는 초보자입니다.

A>
```

는 CRT는 아니고 FILE1에 출력되는 것이다.

"<"는 그후에 쓰여진 화일을 리드모드에서 오픈하여 그 핸들을 표준입력(통상은 키보드)의 화일핸들 0번에 할당한다. 이것은 통상 키보드에서의 입력을 요구하도록 하는 코맨드에 대해서 그것 이외의 입력장치나 화일에서 문자를 입력하도록 지정하는 것이다. 또 ">>"는 화일을 어펜드(Append)모드에서 오픈하는 이외는 ">"과 같다. 즉 이미 존재하는 화일에 코맨드의 출력을 첨가한다.

어펜드모드의 리다이렉션을 이용하면 에디터를 사용하지 않아도 CONFIG.SYS나 AUTOEXEC.BAT화일의 내용 등에 간단히 화일을 첨가할 수 있다. 예를 들면 **리스트 3**에 나타난 바와 같이 TEST1의 화일내용을 TEST2의 화일내용으로 어펜드할 수 있다. 실제로 시험하여 보기 바란다. 단, 이 기능은 매뉴얼에는 나타나 있지 않다. 어펜드모드를 다른 코맨드(예를 들면 COPY 등)에서 사용하였을 경우는 일부 동작이 이상하게 되어 버린다.

아마 동작의 보증을 할 수 없으므로 매뉴얼 등에서 삭제하지 않았을까 생각된다.

●파이프와 필터

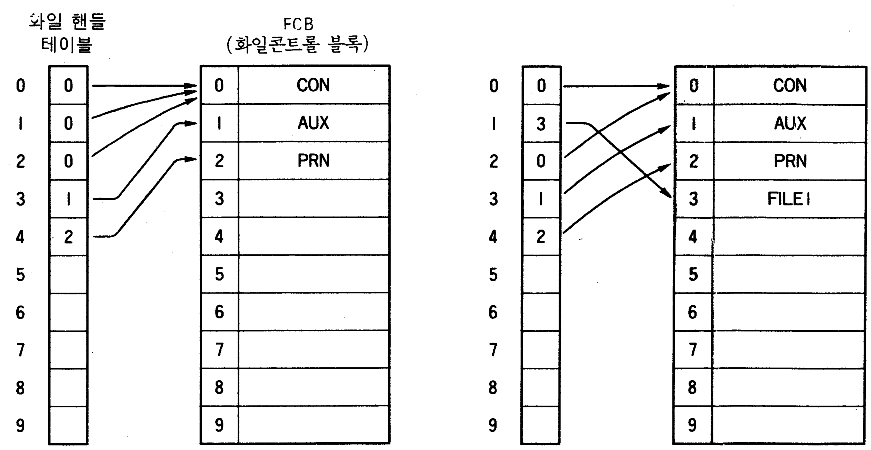
파이프는 표준입출력의 개념에 따라서 실현되며 하나의 강력한 기능이다. 파이프의 기본적인 사고방식은 여러개의 코맨드를 동시에 기동하여 그들의 출력과 입력을 결합하여 버리고자 하는 것이다. 즉 이것은 단순한 코맨드의 조합으로 복잡한 처리를 실행하고자 하는 UNIX적인 아이디어라고 할 수 있다.

UNIX는 본격적인 멀티태스크/멀티유저의 OS이며, 여러개의 코맨드를 동시에 기동할 수 있으므로 파이프처리를 효율좋게 실행할 수 있다. MS-DOS에서는 그것을 할 수 없으므로 템포러리화일(Temporary File)을 작성하여 I/O리다이렉션의 기능에 의해서 이것을 실현하고 있다.

리스트 4의 예에서 보면 중간이 파이프기능을 사용한 디렉토리의 소트예이며, 밑에 있는 것이 I/O리다이렉션을 사용하여 디렉토리의 소트를 하는 경우의 예이다. 결과는 꼭 같은 것이 얻어지지만 파이프에 의해 오퍼레이션을 매우 간략화할 수 있다. 그러나 파이프에서도 결국은 작업용의 템포러리화일을 커런트디렉토리상에 자동적으로 오픈하여 거기에 내용을 써넣어 처리한다. 이 때문에 플로피디스크 시스템에서 큰 화일의 내용을 소트할 경우 등에서는 상당한 실행시간을 요한다.

필터란 간단히 말하면 입력도 출력

〔그림 10〕 I/O 리다이렉션



〔리스트 4〕 파이프에 의한 디렉토리 소트

```
A:\>dir

 Volume in drive A has no label
 Directory of  A:\

ANSI     SYS     1678   3-17-87  12:00p
APPEND   EXE     5825   3-17-87  12:00p
  :
P1                768   3-07-89   2:22p
FILE1            1368   3-07-89   2:25p
        31 File(s)     10240 bytes free

A:\>dir ¦ sort

        33 File(s)     8192 bytes free
 Directory of  A:\
 Volume in drive A has no label
0E1B0D2C            0   3-07-89   2:27p
0E1B0D5E            0   3-07-89   2:27p
  :
SYS      COM     4766   3-17-87  12:00p
VDISK    SYS     3455   3-17-87  12:00p

A:\>dir > temp

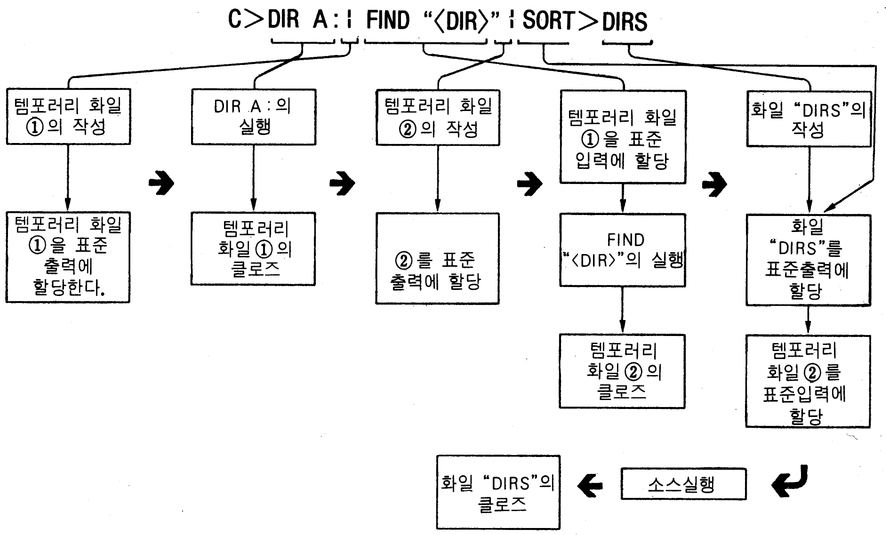
A:\>sort < temp


        32 File(s)     8192 bytes free
 Directory of  A:\
 Volume in drive A has no label
ANSI     SYS     1678   3-17-87  12:00p
APPEND   EXE     5825   3-17-87  12:00p
  :
TEMP                0   3-07-89   2:28p
VDISK    SYS     3455   3-17-87  12:00p

A:\>del temp
```

도 표준입출력을 사용한 코맨드의 경우이다. 요컨데 표준입력에서 파이프를 통해서 입력되어 온 데이터(무엇인가의 코맨드의 출력 등)를 가공하여 다시 표준출력으로 보내는 것이다. 파이프의 기능은 이와 같은 필터의 형식을 가진 코맨드와 조합하여야 비로서 효과를 발휘한다.

〔그림 11〕 파이프의 활동



〔리스트 5〕 필터기능

```
C:\WP\GEMDISK3>dir a:

 Volume in drive A has no label
 Directory of  A:\NORTON

.            <DIR>      3-07-89   2:34p
..           <DIR>      3-07-89   2:34p
PCTOOLS      <DIR>      3-07-89   2:35p
SIDEKICK     <DIR>      3-07-89   2:35p
MASM         <DIR>      3-07-89   2:36p
ADDRESS  EX       1152 11-27-87   1:05a
        6 File(s)     2048 bytes free

C:¦WP\GEMDISK3>dir a:¦ find "<dir>"¦ sort >dirs
```

MS-DOS에서는 표준적인 필터로서 FIND, MORE, SORT 등 3개의 필터 코맨드가 마련되어 있다. 리스트 5의 예와 같이 코맨드를 주면 드라이브A의 디렉토리의 속에서 문자열 "<DIR>"을 포함하고 행, 즉 서브디렉토리명을 포함하는 행을 뽑아 내서 그들을 소트하여 결과를 "DIRS"라는 화일에 라이트한다. 이러한 동작의 모양을 도표로 나타내면 그림 11과 같다.

필터를 사용하여도 한번에 동작하는 코맨드는 1개뿐이므로 그렇게 복잡한 처리가 이루어지는 것은 아니다.

(2) MSDOS.SYS

MSDOS.SYS는 OS의 핵이라고 불리우는 부분이다. OS로서의 기본적인 기능의 대부분은 이 MSDOS.SYS가 중심이 되어 실현되고 있는 것이다. 다른 머신상의 MS-DOS에서도 이 부분만큼은 거의 동일한 것이 사용되고 있다. OS의 기본적 기능으로서의 주변장치에 대한 입출력도 MSDOS.SYS가 통괄하고 있지만 기종에 따라서 다른 디바이스의 종속적인 부분은 나중에 설명하는 IO.SYS가 담당하고 있다.

여기서는 MSDOS.SYS의 중심적 기능인 프로세스관리, 메모리관리에 대해서 설명한 후 어플리케이션 등에서 MSDOS.SYS의 기능을 사용하기 위한 수단인 시스템콜에 대해서 간단히 설명한다.

●프로세스 관리

어플리케이션 등의 외부프로그램이나 MS-DOS시스템의 일부인 COMMAND. COM은 MS-DOS의 아래에서는 하나의 프로세스로서 실행된다. MS-DOS에서는 여러개의 프로세스를 메모리상에 동시에 로드하는 기능(유

감스럽지만 동시에는 실행할 수 없지만)이 있다. 어떤 프로세스가 다른 프로세스를 로드하여 호출하였을 때 원래의 프로세스를 모프로세스, 로드된 쪽의 프로세스를 자프로세스라고 한다.

유저가 프롬프트에 이어서 코맨드를 입력하여 어플리케이션을 기동하는 경우를 생각하여 보자. 이때는 COMMAND. COM이 모프로세스로 되고 어플리케이션 등의 코맨드를 자프로세스로서 메모리에 로드해 이것에 제어를 옮기고 있는 것이다. 자프로세스가 또 자프로세스를 로드하여 호출하도록 하는 것도 가능하다. 이것은 메모리가 허락하는 한 몇겹으로도 할 수 있다.

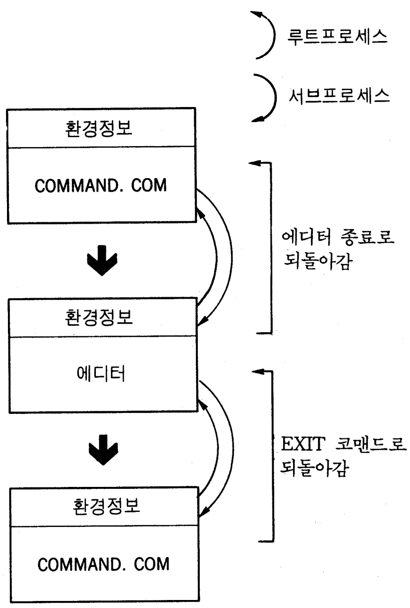
예를 들면 에디터와 같은 프로그램에서는 에디터를 종료시키지 않고 일단 코맨드모드로 되돌려서 MS-DOS의 코맨드를 실행할 수 있도록 하는 것이 있다. 이것은 COMMAND.COM을 모프로세스로 하는 에디터가 또한 COMMAND.COM을 자프로세스로서 로드하여 호출하고 있는 것이다. 이 상태에서는 COMMAND.COM이 모프로세스, 에디터가 자프로세스, 그리고 COMMAND.COM이 다시 자프로세스로 된다. 이때 모프로세스의 COMMAND. COM과 자프로세스의 COMMAND.COM은 전혀 다른 프로세스로서 처리되고, 또한 메모리상에 로드되고 있다. 이것을 **그림 12**에 나타낸다. 이때 어플리케이션에서 기동된 COMMAND.COM은 EXIT코맨드에 의해서 그 모프로세스인 어플리케이션으로 되돌아갈 수 있다. 그렇지만 MS-DOS 기동시에 로드된 COMMAND.COM은 되돌아 가야 할 모프로세스를 가지고 있지 않다. 이 경우는 EXIT코맨드가 실행할 수 없도록 되어 있지 않으면 안된다. 아무 것도 지정하지 않고 부트시에 자동적으로 로드되었을 경우는 이와 같이 되고 있지만 유저가 CONFIG. SYS화일 속의 변수 SHELL에 COMMAND.COM의 패스를 설정하도록 하는 경우에는 **리스트 6**과 같이

〔리스트 6〕 COMMAND.COM의 프로텍트 옵션

```
A>TYPE CONFIG.SYS
SHELL=A:\COMMAND.COM /P ;옵션 /P는 COMMAND.COM을 계속 로드시키며 만약
                         AUTOEXEC.BAT가 있으면 그것을 실행한다.

A>
```

〔그림 12〕 프로세스의 움직임



반드시 "/P"스위치를 붙여서 EXIT 코맨드가 실행할 수 없도록 해 둘 필요가 있다.

모프로세스가 자프로세스를 호출할 때에는 코맨드라인에 주어진 코맨드형이나 파라미터 외에 나중에 설명하는 환경문자열 등의 환경정보가 자프로세스에 계승된다. 그렇지만 자프로세스를 종료하여 모프로세서로 되돌아갈 때에는 자프로세스의 속에 환경이 변경되어도 모프로세서의 환경에는 영향을 주지 않는다. COMMAND.COM을 기동할 수 있는 에디터를 가지는 사람은 한번 시험하여 보면 좋을 것이다.

●메모리관리

MS-DOS에서는 유한한 자원인 메모리를 각 프로세스에 효율좋게 할당하기 위해서 메모리관리가 이루어지고 있다. 외부프로그램은 기동시에 그것이 필요로 하는 메모리를 획득하여 그 범위를 자유로이 사용할 수 있다. 또 프로그램의 동작중에 그것에 할당된 메모리사이즈의 변경도 가능하다. 통상 프로세스의 종료시에는 사용하고 있던 메모리를 해방하게 된다. 이들의 메모리관리에 관한 동작은 모두 시스템콜로서 마련되어 있다.

외부프로그램을 실행하기 위해서는 메모리상에 PSP(Program Segment Prifix)라고 불리우는 영역(256바이트)이 확보되어 실제의 프로그램이 로드되는 것은 PSP의 직후로 된다. 이 PSP의 내용은 모프로세스에서 인도된 환경을 유지하고 있는 환경문자열이 수록되어 있는 환경세그먼트의 어드레스나 코맨드라인에 주어진 문자열 등이다. 여기서 환경문자열은 이른바 ASCIZ 문자열이라고 불리우는 형식으로 메모리에 수록되어 있다. 이것은 C언어에서 취급하는 문자열과 같이 문자열의 널(Null) 문자(00H)가 붙은 형식을 취하고 있다.

●시스템콜

MS-DOS는 MS-DOS상의 어플리케이션을 동작 또는 개발하기 위해서 MS-DOS자신의 내부기능을 사용하는 것을 허락하고 있다. 이것이 일반적으로 이야기되는 시스템콜이라고 불리우는 것이다.

시스템콜의 서포트는 DOS상의 어플리케이션의 개발효율을 높일 뿐만 아니라 DOS자원을 공유화하는 것에 의

한 DOS어플리케이션의 기종 의존성의 회피, 어플리케이션 프로그램자체의 소형화를 실현할 수 있다.

시스템콜은 8088/86CPU의 소프트웨어인터럽트의 기능을 이용하여 실현되고 있다. 통상 인터럽트라고 하면 외부인터럽트를 가리킨다. 이것은 CPU 외부에서의 신호에 대해서 CPU가 응답하여 그때 하고 있던 동작을 중단하고 그 신호에 대해서 정해진 처리를 수행하는 것이다. 인터럽트와 그것에 대응하는 루틴(인터럽트 서비스루틴)의 어드레스는 인터럽트 벡터테이블이라고 불리우는 메모리상의 영역에 기록되어 있다.

한편 소프트웨어인터럽트는 프로그램 속에 묻어 넣어진 명령에 따라서 인터럽트를 발생하여 그것에 대응하는 서비스루틴을 기동한다. 이것은 서브루틴콜과 유사하지만 서브루틴을 호출할 때에는 그 어드레스를 지정하지 않으면 안된다. 한편 소프트웨어인터럽트에서는 머신에 따라서 달라지고 있는 서비루틴의 어드레스를 알지못해도 그것을 호출할 수가 있다.

8088/86CPU에는 00H~FFH까지의 번호가 붙여진 256종류의 인터럽트 서비스루틴을 선택하는 기능이 있다. 이 가운데 20H~FFH가 소프트웨어인터럽트용이다. 그중에서 MS-DOS의 외부프로그램이 사용할 수 있는 것은 20H, 21H, 25H, 26H, 27H로 되어 있다. 이 가운데서도 21H는 시스템콜이라고 불리우며 가장 중요한 역할을 하고 있다.

시스템콜에는 어플리케이션이 MS-DOS의 기능을 최대한으로 이용하기 위한 가지각색의 것이 마련되어 있다.

(3) IO.SYS

MSDOS.SYS와 하드웨어의 사이에 넣어서 실제의 입출력을 담당하고 있는 것이 IO.SYS이다. IO.SYS에 의해서 머신마다의 다른 하드웨어규격을 흡수하고 있는 것이다. MSDOS.SYS는 기종에 의한 차이가 거의 없는데 비해 IO.SYS는 기종에 따라서 크게 다르다.

IO.SYS의 내용은 그 기종의 각 주변장치에 마련된 디바이스드라이버라고 불리우는 프로그램모듈을 모아놓은 것이다. 이 디바이스드라이버에는 1바이트마다의 입출력이 가능한 캐릭터형 디바이스(디바이스화일)를 취급하는 것과 몇 바이트인가를 종합하여 취급하는 블록형 디바이스(드라이브)을 취급하는 것으로 크게 나누어진다.

●디바이스드라이버

MS-DOS의 표준디바이스드라이버는 CON(콘솔), AUX(RS-232C 등), PRN(프린터), 클럭과 디스크의 5종이다. 이 가운데 CON, AUX, PRN, 클럭의 4개는 캐릭터형이고, 디스크드라이브만이 블록형디바이스이다. 또 CON, AUX, PRN의 3개에 대해서는 디바이스명을 그대로 논리화일명으로서 사용할 수 있기 때문에 MS-DOS의 코맨드라인상에서 통상의 화일명과 전부 동등하게 취급한다. **리스트** 7은 CON디바이스를 사용한 예이다. COPY코맨드는 콘솔에서 입력된 문자를 PARK1이라는 화일에 카피한다. 입력의 종료에는 CTRL+Z를 입력한다.

통상은 1개의 드라이버로 1종의 I/O장치를 취급하지만 블록형은 예외이며, 예를 들면 플로피디스크 드라이버와 같이 1개의 드라이버로 여러개의 기기(A, B 양 드라이브나 2DD나 2HD 등의 다른 타입의 미디어)를 취급할 경우도 있다.

상기한 표준이외의 디바이스드라이버도 CONFIG.SYS화일에 등록함으로써 간단히 추가등록할 수 있다. 예를 들면 **리스트** 8과 같이 마우스드라이버를 등록하는 예가 자주 눈에 띈다. 개개의 디바이스드라이버는 다른 드라이버에 대한 링크정보를 가지고 있다. 새로운 디바이스드라이버의 추가는 이 링크정보를 바꾸어 씀으로서 쉽게 할 수 있는 것이다. 캐릭터형 디바이스의

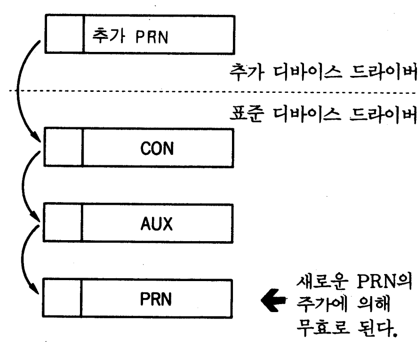
〔리스트 7〕 CON디바이스에서의 입력

```
A>COPY CON PARK1
월간 퍼스널컴퓨터는
1988년 12월에 창간된
비전문인을 위한
컴퓨터전문지입니다.
^Z

A>TYPE PARK1
월간 퍼스널컴퓨터는
1988년 12월에 창간된
비전문인을 위한
컴퓨터전문지입니다.

A>
```

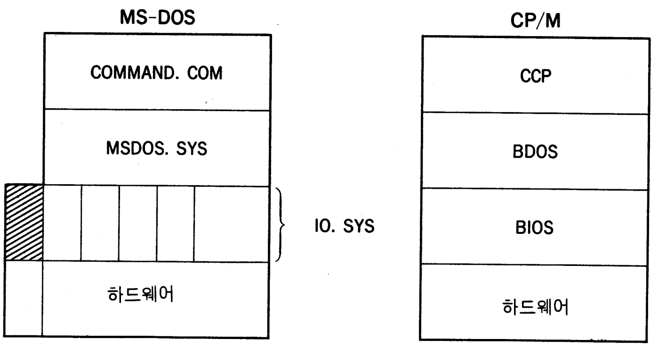
〔그림 13〕 캐릭터형 디바이스 드라이버의 추가



〔리스트 8〕 디바이스 드라이버의 등록

```
A>TYPE CONFIG.SYS
DEVICE=MOUSE.SYS ;주변장치 사용을 위한 디바이스 드라이버
                  프로그램을 지정한다.
A>
```

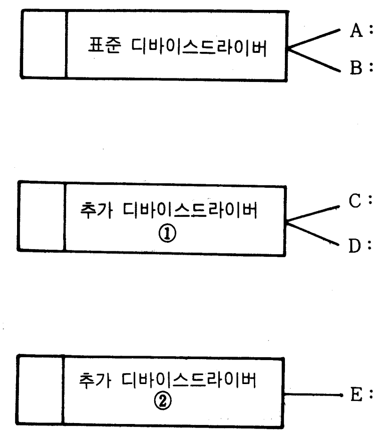
〔그림 14〕 MS-DOS 디바이스드라이버와 CP/M



하드웨어의 확장에 대해서도, 이것에 사용되는 디바이스 드라이버의 추가로 대응한다.

CP/M의 BIOS는 고정적인 것이고 하드웨어를 확장할 때는 BIOS, 경우에 따라서는 그 외의 부분도 만들어 바로잡을 필요가 있다.

〔그림 15〕 블록형 디바이스드라이버의 추가



경우 새로운 드라이버는 원래의 링크 앞에 추가된다. 이것을 그림으로 나타내면 **그림 13**과 같다. 이때 이전부터 있었던 것과 같은 드라이브명을 가지는 드라이버를 등록하면 기존에 있는 드라이브명은 액세스되지 않게 된다. 이것에 의해서 예를 들면 유저가 자기가 가진 기종전용의 프린터드라이버를 사용하고 싶은 경우에는 새로운 PRN 이라는 이름의 드라이버를 등록하면 되는 것이다.

디바이스드라이버에는 표준적인 입출력의 순서가 정해져 있어서 이 규칙에 따르는 것이면 모두 MS-DOS에 의해서 서포트된다. 그러므로 어플리케이션프로그램이나 MS-DOS본체에 아무런 변경을 가하는 것 없이 다른 하드웨어에 별문제없이 대응할 수 있는 것이다. **그림 14**는 MS-DOS와 CP / M의 구성을 나타낸 것이다. CP / M에서는 디바이스드라이버에 상당하는 부분은 모두 BIOS로 꾸며 넣어지기 때문에 새로운 디바이스를 서포트하고자 하면 BIOS(MS-DOS의 IO.SYS에 상당하는 부분)라고 불리우는 부분은 새삼스럽게 다시 작성하지 않으면 안되었다. 이 당시에도 MS-DOS의 우위성과 시스템의 유연성의 일단을 볼 수 있다. 그러나 CP / M은 소스가 유저에게 공개되어 있기 때문에 시스템을 충분히 알고 있는 유저는 직접 BIOS를 조작하여 확장할 수 있다. MS-DOS는 OEM공급이라는 형태를 취하기 때문에 유저레벨에서도 확장가능하도록 디바이스드라이버라고 하는 형이 취해지고 있다고도 해석할 수 있다.

●드라이브(블록형 디바이스)와 BPB

블록형디바이스란 각종 디스크에 대표되도록 램덤액세스가 가능한 것을 가리킨다. 이들은 캐릭터형디바이스와는 달라서 임의의 이름은 아니고 "A:", "B:", "C:"와 같은 드라이브명으로 식별한다. 새로이 드라이브를 추가하였을 경우는 그것 이전에 있었던 드라이브명 다음의 알파벳 이름이 자동적으로 붙여진다.

블록형디바이스의 경우 **그림 15**에 나타내는 바와 같이 추가된 디바이스드라이버는 이전에 존재하고 있었던 것의 뒤에 링크되기 때문이다.

하드웨어적 성격이 다른 디스크를 원활하게 또는 동등하게 취급하기 위해서 각각의 드라이브에서 취급하는 미디어의 종류마다에 BPB(Bios Parameter Block)라고 불리우는 일종의 표가 디바이스드라이버내에 마련되어 있다. BPB에는 표 2에 나타낸 바와 같은 디스크의 포맷과 동등한 그 미디어의 성격이 기록되어 있다. 디바이스드라이버는 그 드라이브에 삽입된 미디어의 FAT ID를 패치함으로써 그 종류를 식별하여 BPB의 내용을 MSDOS. SYS에 넘겨준다. 이것에 의해서 MS-DOS. SYS는 미디어의 종류에 따른 처리를 실행할 수 있는 것이다.

MSDOS.SYS는 이와 같은 미디어체크기능에 의해서 어느 드라이브에 어느 타입의 미디어가 세트되어 있는가를 항상 파악하고 있는 것이다.

프로그램의 실행

앞에서 설명한 바와 같이 MS-DOS에서는 IO.SYS와 MSDOS.SYS, COMMAND.COM의 3가지의 기능을 분담시키고 있다. 왜 이와 같은 귀찮은 것을 하는 것일가? 여기에는 충분한 이유가 있다.

만일 이 3개가 분리되어 있지 않는다고 하면 어떻게 될 것인가? 우선 프로그램이 커진다. 커진 프로그램이라는 것은 그만큼 내부에 버그를 포함할 가능성이 커진다는 말이 된다. 또한 버그를 발견하고자 하여도 어디에 있는지 찾는 것이 어렵게 되어 프로그램을 전부 바꾸어야 하는 경우가 생긴다.

이러한 문제를 회피하기 위해서 직

접 하드웨어를 움직이는 부분은 IO.SYS에, MS-DOS의 기본적인 기능은 MS-DOS. SYS에, 유저의 입력을 해석하는 부분은 COMMAND.COM에 분담시키고 있는 것이다. 또 단지 기능을 분담시키는 것 만이 아니고 각각의 프로그램을 호출하는 방법을 통일하고 있다. 어셈블리레벨에서 MS-DOS의 펑션콜을 이용할 때에는 소프트웨어인터럽트의 21H를 사용하지만 이것은 틀림없이 유저프로그램 입장에서 보면 MS-DOS. SYS를 호출하고 있는 것이다.

또한 MSDOS.SYS에서도 실제로 플로피디스크 등을 리드·라이트할 때는 IO.SYS를 호출하고 있다. 이와 같이 호출방법을 정확하게 결정하여 두면 호출되는 쪽의 내부처리가 바뀐 곳에서는 호출하는 쪽의 처리에 영향을 주지 않게 된다. 따라서 이들 3개의 프로그램을 따로따로 테스트할 수 있으므로 개발의 수고는 상당히 줄게 된다.

COMMAND.COM의 메모리배치는 상당히 특수한 것으로, 상주부와 비상주부라고 불리우는 2개의 부분으로 나누어져서 주기억상에 배치된다. 그리고 8088CPU의 특성을 살려서 각각을 다른 세그먼트로서 취급하고 있다.

우선 초기화루틴이나 외부코맨드를 기동하는 부분 등을 핵으로 한 부분이 메모리의 하위번지에 위치되어 상주부라고 불리운다. 이것을 상주부라고 부르는 것은 코맨드에 제어가 이동하고 있을 때에도 항상 메모리 상에 존재하기 때문이다.

상주부에 대해서 비상주부라고 불리우는 부분은 COMMAND.COM자신이 가지고 있는 실행루틴(내부코맨드도 포함한다)을 중심으로 구성되어 메모리의 최상위부에 배치된다. 비상주부가 기억되어 있는 부분은 외부코맨드로 제어가 이동하고 있을 때에도 외부코맨드를 자유로이 사용할 수 있다. 그러므로 외부코맨드에서 COMMAND.COM으로 제어가 되돌아 간 시점에서 비상주부가 바꾸어 쓰여지고 있지 않은가 어떤가의 체크를 하여 외부코맨드에 의해서 바꾸어 쓰여지고 있을 경우에는 상주부를 다시 로드하도록 되어 있다.

왜 COMMAND.COM이 이와 같은 복잡한 처리를 하고 있는 것인가? 이것은 MS-DOS가 개발된 당시의 하드웨어에 기인한다고 생각된다. 지금은 대부분의 기종이 640K바이트 이상의 메모리를 내장한 채 시판되는데 MS-DOS가 개발된 당시는 메모리칩의 칩당 용량이 적고 또한 칩자체의 가격이 높기때문에 표준실장하는 메모리용량은 128K바이트 정도였다.

메모리가 128K바이트이어도 최소의 필요한 프로그램(MSDOS.SYS나 IO.SYS, 디바이스드라이버 등)을 위해서 메모리를 공제하게 되면 70K바이트 정도 밖에 남지 않는다. 그래서 될수록 외부코맨드에서 사용할 수 있는 메모리를 증가하기 위해서 아무래도 필요한 부분만을 상주부로 하여 외부코맨드로 제어가 이동하였을 때에는 필요가 없는 부분을 비상주부로서 적은 메모리를 어플리케이션프로그램으로 해방할 필요가 있었던 것이다.

이와 같은 노력을 기울여도 MASM.EXE(어셈블러)나 LINK.EXE(링커)라고 하는 프로그램개발용의 외부코맨드는 128K바이트의 메모리에서는 실행할 수 없다. 또한 현재의 컴퓨터는 대부분 640K바이트이상의 메모리를 표준실장하고 있으므로 상주부, 비상주부라고 하는 개념도 사실상 필요없게 되어 버렸다.

IBM은 자사의 PS/2라고 불리우는 80286/80386CPU를 탑재한 머신에 MS-DOS 버전5.0을 베이스로 한 OS/2의 서포트를 명확하게 하고 있다. OS/2에 있어서도 COMMAND.COM은 상주부와 비상주부로 구성되어 있지만 이것은 MS-DOS 버전3.1용의 S/W를 계승하기 위해서이며, 80286/80386 CPU의 프로텍트모드로 동작하는 전용의 COMMAND.COM에 상당하는 셀이 별도로 마련되어 있다. 이 셀에서는 16M바이트라는 광대한 메모리공간을 풀로 구사하는 것이 가능하며, 상주부, 비상주부라고 하는 구분은 없다.

(1) 코맨드의 입력

코맨드해석, 실행루틴은 COMMAND.COM의 비상주부에 있다. 여기서는 펑션콜의 10번을 사용하여 콘솔에서 1행 읽어 넣는다. 다음에 이후의 작업에 준비하여 이것을 FCB에 카피하여 둔다. FCB는 File Control Block의 약자이며, 화일에 액세스할 때 필요하게 되는 정보를 종합하여 기억해 두기 위한 영역이다. FCB로 사용하는 영역은 미리 MS-DOS의 펑션콜을 호출하는 측에서 마련하여 둘 필요가 있다.

그후는 펑션콜의 41번을 사용하여 화일명을 해석한다. 그 결과 코맨드로서 주어진 문자열에 와일드카드가 사용되고 있기도 하고, 드라이브의 지정이 무효였을 경우는 각각의 에러표시를 하여 1행 읽어 넣는 부분으로 되돌아 간다.

드라이브의 지정이 유효하여 와일드카드도 사용되고 있지 않으면 입력된 문자열을 FCB에 전송한다. 즉 유저가 COMMAND.COM의 내부코맨드테이블을 변경하여 독자의 코맨드를 정의한 경우 제2인수 이상을 필요로 하는 것 같은 파라미터의 수수는 할 수 없게 된다.

입력된 것이 빈행일 때는 아무 것도 시키지 않고 메인루프도 되돌아 간다. 즉 스페이스를 몇문자입력하여도 COMMAND. COM은 접수하여 주지 않는 것이다.

(2) 코맨드의 실행

입력된 문자가 코맨드로서 인식되면 COMMAND.COM은 이것을 실행하려고 한다.

COMMAND.COM은 코맨드를 실

행할 때 우선 내부코맨드의 검색을 한다. 만일 내부코맨드에 입력된 문자와 일치하는 코맨드가 없을 경우는 외부코맨드를 검색한다. 즉 내부코맨드와 같은 이름의 외부코맨드를 만들어 두어도 실행은 할 수 없게 된다.

따라서 내부코맨드와 같은 이름의 외부코맨드는 확장자명을 무엇을 붙여도 무의미한 것이 된다.

(3) 내부코맨드

내부코맨드는 최초에 코맨드의 문자수+1의 숫자가 있고, 다음에 코맨드명이 대문자로 기재되어 있다. 코맨드명의 다음 1바이트는 인수에 와일드카드를 사용할 수 있는지 어떤지를 나타내고 있다. 이것이 1이면 와일드카드를 사용할 수 있고, 0이면 와일드카드를 사용할 수 없는 것을 나타내도록 되어 있다.

즉, COMMAND.COM내부의 코맨드명 테이블의 체크가 종료되면 내부코맨드인가 외부코맨드인가에 따라 처리가 분기하여 온다.

입력된 코맨드가 내부코맨드테이블에서 발견하였을 경우 아까 체크한 인수에 사용된 드라이브명의 타당성을 조사한다. 여기서 드라이브명이 무효이면 에러메시지를 출력하여 메인루프로 되돌아 간다. 드라이브명 및 인수가 유효한 경우는 코맨드테이블에 기술되어 있다. 실행루틴의 오프셋어드레스를 서브루틴으로서 콜하여 실제의 처리를 한다. 처리가 종료하면 1행 읽어 넣는 부분으로 되돌아 간다. 여기서 MS-DOS는 프롬프트를 표시하여 유저에게 코맨드의 입력이 가능한 것을 촉구하고 입력대기상태로 된다.

내부코맨드는 어느 코맨드가 실행되

어도 처리하는 서브루틴어드레스가 변화할 뿐이며 기본적인 처리의 흐름은 전혀 변함이 없다.

(4) 외부코맨드

내부코맨드테이블을 검색하여 내부코맨드에 없는 문자열이였을 경우는 내부코맨드 실행루틴에서 빼내서 외부코맨드의 실행루틴으로 제어가 이동한다.

외부코맨드의 실행순서는 확장자명(COM, EXE, BAT) 때문에 내부코맨드의 검색루틴에 비해서 다소 복잡한 처리로 되어 있다.

처음에 디렉토리엔트리를 검색하기 위한 이름부분에 주어진 코맨드를 넣어 확장자의 부분을 "?"로 채운다. 이 상태에서 디렉토리엔트리를 검색하면 확장자에 관계없이 입력된 코맨드와 같은 이름을 가진 화일을 검색한다.

즉 최초에 화일명의 타당성을 체크하고 다음에 확장자명의 타당성을 체크하는 2중의 화일검색을 하고 있는 것이다.

검색의 결과 화일이 발견되고, 그 화일의 확장자가 COM이였을 경우는 곧 그것을 실행한다. 확장자가 EXE 또는 BAT의 화일이 최초에 발견되었을 경우는 각각의 확장자명을 나타내는 플래그를 오프 3019H로 세트하고 검색을 계속한다. 만일 디렉토리엔트리가 발견되지 않게 된 상태에서 실행가능한 화일이 발견될 경우는 우선순서(COM→EXE→BAT)에 따라서 실행한다.

이 결과에서 COM확장자명 화일은 될수록 디렉토리엔트리의 최초에 배치하도록 하는 것이 디스크 액세스에 있어서의 처리속도를 단축할 수 있게 된다. EXE와 BAT화일의 경우는 디렉토리엔트리의 도중에서 발견되어도 반드시 디렉토리엔트리의 최후까지 검색하므로 엔트리중에 어디에 배치되어 있어도 검색속도에는 영향을 주지 않는다. 하드디스크 등에서 하나의 디렉토리내에 많은 화일을 수록할 경우 확장자가 COM인 화일을 디렉토리엔트리의 선두부터 배치하는 것이 바람직스러운 방법이다. 또 COM 확장자의 화일만을 하나의 디렉토리내에 놓아버리는 등도 유효한 방법으로 될 것이다.

만일 실행가능한 화일이 없을 경우 커런트디렉토리를 환경변수 "PATH"로 지정된 순서에 따라서 검색을 계속한다.

●COM, EXE형식 프로그램의 실행

COM, EXE형식의 외부프로그램은 8088기계어로 기술되어 있어서 COMMAND.COM에서 직접 기동할 수 있다.

COM형식과 EXE형식의 차이를 이해하기 위해서는 8088이라는 CPU의 구조를 알지 않으면 안된다.

8088에서는 주기억용량을 1M바이트까지 가질 수 있도록 되어 있다. 1M바이트는 2의 20승이므로 단순히 생각하면 주기억상의 특정한 1바이트를 지정하기 위해서는 20비트의 어드레스를 필요로 한다는 말이 된다. 그러나 8088의 레지스터세트를 보아도 20비트의 레지스터는 없고, 내부레지스터는 최대가 16비트이다. 16비트의 어드레스로 1M바이트의 주기억을 지정하는 방법으로서 생각해 낸 것이 『세그먼트』라는 개념이다.

1개의 세그먼트는 64K바이트의 크기를 가지고 있어서 64K바이트마다 시작되는 것을 지정할 수 있다. 나중에 세그먼트의 시작된 곳에서 바이트수를 지정하면 되는 것이다. 이렇게 하면 세그먼트와 오프셋이라는 2개의 16비트 어드레스로 1M바이트 모두를 지정할 수 있는 것이다. 그림 16과 같이 세그먼트를 지정하는 16비트를 왼쪽으로 4비트 시프트하고 나서 하위4비트에 0을 넣고, 오프셋을 더하여 주면 실제의 어드레스가 구해지는 것이다.

8088의 레지스터중 세그먼트레지스터라고 불리우는 4개의 레지스터가 이 세그먼트의 개시번지를 기억하는 것으로 되어 있다. 기계어의 명령은 CS레지스터에서 지정되는 세그먼트에 없으면 안된다. 데이터는 DS레지스터에서 지정되는 세그먼트에 있다고 하는 전제에서 실행되지만 변경할 수도 있다.

전송명령 등의 전송세그먼트는 ES레지스터에서 지정하여 두지 않으면 안된다.

〔그림 16〕 8088에서의 어드레스 계산법

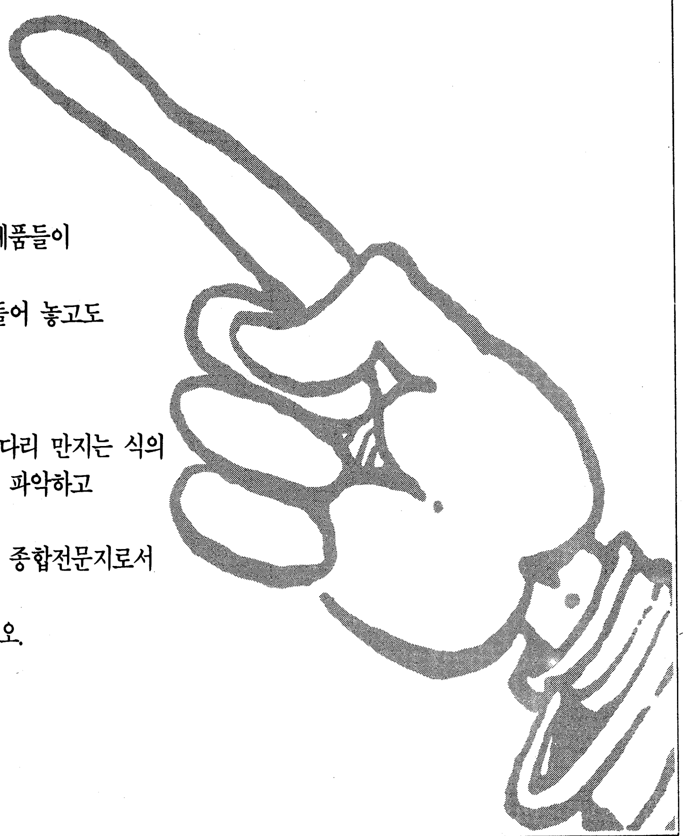
```
  1000H    ← 세그먼트
+ 0100H    ← 오프셋
 10100H    ← 액세스되는 실제 어드레스
```

COM형식의 외부프로그램은 임의의 세그먼트 오프셋 100H에서 로드, 실행된다고 하는 전제하에 화일의 선두부터 직접 기계어로 명령이 쓰여지고 있다. 외부기억장치에서 주기억상으로 전송할 때의 처리는 간단하다. 또 프로그램실행개시시에는 모든 세그먼트레지스터에 같은 값이 들어 있어서 오프셋만을 생각하면 되므로 어드레스에 관해서는 8080이나 Z80 등의 8비트 CPU 와 같은 감각으로 취급할 수 있다. 이 때문에 COM형식의 것을 8080 모델이라 부르기도 한다. 반대로 오프셋만 취급하지 않기 때문에 프로그램과 데이터를 합해서 64K바이트 이내에 수록할 필요가 있다.

소규모인 프로그램이라면 어쨌든 8088이 가지는 1M바이트의 주기억을 활용하려고 생각하였을 때에 COM모델은 1세그먼트 이내라고 하는 제한에 따라 대량의 메모리를 최대한으로 이용할 수는 없다.

EXE형식의 프로그램은 여러개의 세그먼트 집합체의 형을 갖는다. 이 때문에 실제로 주기억상에 배치되기까지 세그먼트의 상호관계가 결정될 수 없으므로 어드레스의 정보를 상대적으로 가지고 있어서 실제로 배치될 때에 이것을 결정한다고 하는 복잡한 작업이 필요하게 된다. 이것 이외에도 EXE

화일인 것을 나타내는 식별자나 최저한 필요로 하는 주기억용량, SS / SP 레지스터의 초기값 등의 정보가 필요하기 때문에 EXE화일은 선두가 갑자기 실행가능한 기계어로 되어 있는 것은 아니고 대신에 이들의 정보가 들어있는 것이다. 이들의 정보를 분석해서 처리하기 위해서 주기억상에 배치하는데에 약간 시간을 요하지만 EXE형식에서는 8088의 능력을 활용할 수 있다.



키보드의 스캔코드 값을 구한다

IBM PC의 키보드는 훌륭한 유연성을 지니고 있어서 프로그래머가 소프트적인 제어방법으로 모든 키를 할당할 수 있다. 이는 키보드의 각 문자세트와 함수 기능들이 키(key)와 물리적으로 연결되어 있지 않고 모두 스캔코드(Scan Code)로 연결되어 있기 때문에 가능한 것이다.

이러한 유연성으로 인해 우리 주위의 IBM PC 또는 호환기종에서 구동되고 있는 각종 패키지들을 보면 제어, 함수키들이 다양하게 할당되어 있어서 그만큼 폭넓은 기능으로 사용자와의 인터페이스를 마련해 주고 있다.

소프트적인 방법으로 키를 제어하기 위해서는 반드시 해당 키의 스캔코드를 알아야 하는데 IBM PC 매뉴얼(IBM Technical Reference)에 어느 정도의 스캔코드 리스트가 나와 있기는 하지만 불완전한 것이다. 그래서 한번쯤 유용한 프로그램을 작성하려고 도전했다가 〈Alt〉키를 이용한 조합키의 스캔코드값(DOS매뉴얼에는 나와 있지 않음)을 몰라 고민을 하는 독자들이 있을지도 모른다.

리스트 1의 KEYSCAN.ASM은 입력되는 키에 해당하는 스캔코드와 아스키 코드(ASCII Code)값을 출력시키는 프로그램이다.

출력은 XXYY형태로 표시되며 XX는 스캔코드의 16진수형을, YY는 아스키 코드값을 나타낸다(화면 1 참조). 그리고 프로그램을 끝내고 싶으면 〈C-trl〉-〈Brk〉를 누르면 된다.

메인 루틴에서 키보드로부터 키입력

〔리스트 1〕 KEYSCAN.ASM

```
;*******************************************************
;*
;*          입력된 키의 스캔코드와 ASCII코드를 찾기 위한 프로그램
;*          <Ctrl>-<Brk>를 누르면 프로그램이 중지됩니다.
;*
;*******************************************************

ASMUTIL  SEGMENT
         ASSUME    CS:ASMUTIL, DS:ASMUTIL

         ORG  100H

KEYSCAN  PROC
         JMP PASTDATA
KEYCODE  DB  ?,?,?,?,0DH,0AH,'$'
MSG      DB  'Strike any key, Ctrl-Brk to exit',0DH,0AH,'$'
PASTDATA:
         MOV  DX,OFFSET MSG        ;메시지의 오프셋번지
         MOV  AH,9                 ;화면에 메시지 출력
         INT  21H

GETKB:   SUB  AX,AX                ;키입력을 위해 AX 조정
         INT  16H                  ;키 입력 대기
         MOV  DI,OFFSET KEYCODE    ;캐릭터 영역 지정
         CALL AX2HEX
         MOV  DX,OFFSET KEYCODE    ;16진수 형태로 출력
         MOV  AH,9
         INT  21H
         JMP  GETKB                ;새로운 입력키를 위해
                                   ;GETKB로 점프
KEYSCAN  ENDP
ASMUTIL  ENDS

ASMUTIL  SEGMENT  BYTE PUBLIC 'CODE'
         ASSUME  CS:ASMUTIL
AX2HEX   PROC     NEAR
         PUBLIC   AX2HEX
         CLD               ;스트링을 읽어 나감
         PUSH  BX          ;레지스터 저장
         PUSH  CX
```

```
         PUSH  DX
         MOV   BX, OFFSET HEX ;16진수 캐릭터의 오프셋번지
         MOV   CX,4           ;CX에 4 세트(동작 횟수)
NEXTNIBL:
         ROL   AX,1       ;1비트 좌회전
         ROL   AX,1       ;1비트 좌회전
         ROL   AX,1
         ROL   AX,1
         MOV   DX,AX      ;DX에 저장
         AND   AL,0FH     ;하위 바이트(AL)를 분리
         XLAT  CS:HEX     ;16진수 형으로 변환
         STOSB            ;16진수형 출력준비
         MOV   AX,DX      ;다시 AX의 값을 얻음
         LOOP  NEXTNIBL   ;반복 루프(4번)
         POP   DX         ;스택에서 2바이트를 복원하고
         POP   CX         ;리턴
         POP   BX
         RET
AX2HEX   ENDP
HEX      DB  '0123456789ABCDEF','$'
ASMUTIL  ENDS
         END  KEYSCAN
```

〔화면 1〕 KEYSCAN.COM 실행

```
C:\MASM>KEYSCAN
Strike any key, Ctrl-Brk to exit
1E41 ┐
3042 │
2E43 ├ 'A'~'G'
2044 │
1245 │
2146 │
2247 ┘
1C0D  ←<ENTER>키
3B00 ┐
3C00 │
3D00 │
3E00 │
3F00 │
4000 ├ <F1>~<F10>
4100 │
4200 │
4300 │
4400 ┘
0^C   ←<Ctrl>-<Brk>를 눌렀다.

C:\MASM>
```

을 기다리기 위해 INT 16H를 수행하고 임의의 키가 입력되면 그 키에 해당하는 스캔코드와 아스키 코드로 변환시키는 루틴을 호출하고 있다. 이 루틴(AX2HEX)에서는 두 코드의 값을 각각 0~F 범위를 갖는 4개의 16진수 값으로 되돌린다. 이들 변환된 수치들은

```
MOV  DI,OFFSET KEYCODE
MOV  AH,9
INT  21H
```

에 의해 화면에 디스플레이 되고 KEYSCAN 루프는 다음에 입력되는 키를 위해 다시 처음으로 점프된다.

```
JMP  GETKB
```

참고로, 〈Ctrl〉-〈Brk〉는 DOS에 의해 자동적으로 처리되므로 이 조합코드의 스캔코드만을 얻을 수 없음을 말해 둔다. 프로그램 실행환경은 다음과 같다.

먼저 프로그램 에디터나 워드프로세서 등으로 소스화일을 작성한 후 매크로어셈블러(MASM)로 어셈블 한다.

```
C:\>MASM KEYSCAN
```

어셈블이 완료되면 링크프로그램(LINK) 으로 즉시 실행가능한 .EXE 화일을 만든다. 이때 "Missing Stack Segment"라는 메시지가 나타나는데 나중에 EXE2BIN프로그램을 이용하여 .COM화일을 만들 예정이므로 이 메시지는 무시해 버려도 좋다. PC

```
C:\>LINK KEYSCAN
C:\>EXE2BIN KEYSCAN KEYSCAN.COM
C:\>
```

LX-800의 버그

심성호/명지대학교 P.R.D.A.

LX-800에서 그래픽을 하드카피 할 경우 그림과 그림 사이에 한 도트 정도가 빠져버리는 현상이 생긴다. 이러한 버그는 프린터 제어 인터럽트로 들어오는 데이터를 체크하여 바꾸어 주면 간단히 해결할 수 있다.

얼마전까지만 해도 프린터(9핀 프린터의 경우)는 약 50만원정도는 주어야 살 수 있는 고가품으로 생각했다. 그러나 현재 LX-800 프린터는 여러가지 이유로 무척이나 저렴한(?)가격으로 구입할 수 있게 되었다.(참고로 삼보 컴퓨터에서는 LX-800 프린터를 39만원선에서 판매를 하고 있고 외관이나 그 기능이 LX-800과 같은 대우전자의 CPP-1500 프린터는 43만원에 그리고 학생들이 많이 사용하고 있는 용산전자 상가나 청계천의 EPSON AMERICA, INC의 LX-800 프린터는 33~35만원선에서 판매되고 있다.)

하지만, 아직도 학생의 신분으로는 구입하기가 어려운지도 모른다. 하여간 이렇게하여 구입한 프린터가 100% 활용이 되지 않는다면 프린터는 단지 장식물에 지나지 않는다고 할 수 있다.(현재 그렇게 사용하는 사람이 많은 것으로 안다.) 더우기 현재 우리나라에 들어와 있는 LX-800은 국내회사의 제품보다는 미국에서 수입된 프린터가 더 많은 실정이다.(필자의 프린터도 EPSON AMERICA, INC의 제품임)

여기에서는 LX-800 프린터의 새로운 기능이나 중요한 사용법 그리고 하드카피시 생기는 줄을 제거하는 방법등을 소개하고자 한다.

LX-800 프린터는 일본의 엡슨사에서 제작한 것으로 현재 사용하고 있는 프린터 중에서 가장 호환성이 높은 프린터이다. 심지어는 다른 프린터 제조회사도 엡슨사의 제어코드를 따르고 있는 실정이다. 또한 시중에 나와있는 9핀 프린터 중에서는 인자속도가 가장 빠르다고 한다. 물론 속도가 빠르다고 해서 다 좋은 것이라고는 볼 수 없다. 500 CPS(Character Per Second) 정도가 되면 헤드가 빨리 열을 받고 쉽게 손상이 된다. LX-800 프린터의 또다른 장점은 다운 로드(Down Load)기능을 추가하였다는 것이다. 일반적으로 (예를 들면 이용자가 필요로 하는) 문자를 사용할 경우 이 방법을 사용하면 무척 편리하다. 마지막으로 LX-800 프린터는 아직 9핀 프린터에서는 보지 못했던 Auto Load기능이 있다. 즉 종이를 프린터에 삽입할 때 자동으로 LOAD하여 준다.

지금까지 LX-800 프린터의 칭찬만 하였는데 LX-800 프린터도 다 좋은 것은 아니다. 엡슨사에서 고의적인 것이었는지 아니면 콘트롤러에 문제가 있는지는 잘 모르겠지만 하드카피를 할 경우 그림과 그림사이에 한 도트 정도가 빠져 버리는 현상이 생긴다.(그렇지 않은 것도 있음) 그런데 대부분의 사람들은 이러한 사실을 잘모르고 있는 듯 하다. 뒤에서 다시 설명하겠지만 이러한 오류는 간단히 램상주 프로그램으로 바로 잡을 수 있다.

다운로드기능에 대하여

●다운로드란 무엇인가?

다운로드(Down Load)는 사용되어 지고 있는 글씨체(Draft, Italic, International, Character Graphics, and NLQ) 이외에 사용자가 필요한 그림을 프린터에 전송하여 일반 캐릭터와 동일한 방식으로 사용하는 것을 말한다. LX-800 프린터는 Draft와 NLQ(Near Letter Quality)의 모드에서 다운로드할 수 있다.

만약 당신이 필요한 문자를 꼭 사용하여야 할 경우(한 예로 상표를 인쇄할 경우)가 생기다면 어떻게 하겠는가. 다운로드는 이럴때 필요한 문자를 정의할 수 있다.

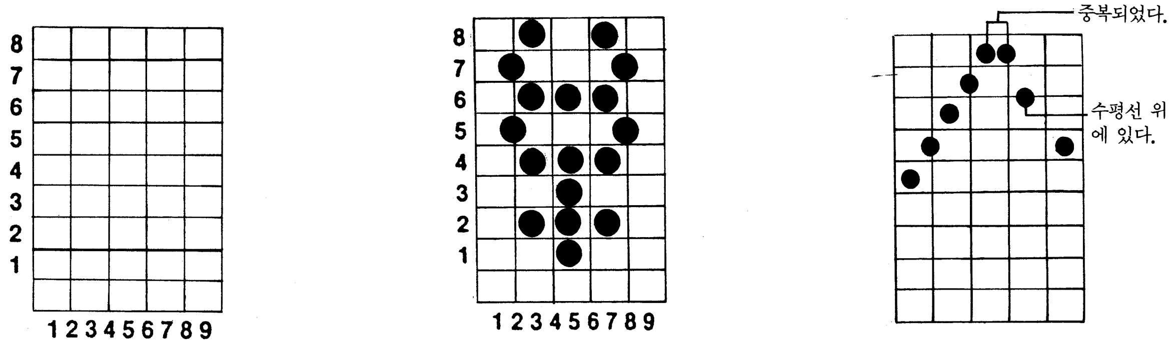
●다운로드의 장점과 단점

다운로드를 사용하면 빠른 속도로 인자할 수 있다. 하지만 그 사용자 패턴을 디자인한다는 것도 어렵거니와 일반 캐릭터와 치환이 되는 것이므로 일반 캐릭터를 사용할 경우에는 모드를 바꾸어야 하는 번거로움이 있다. 또 다운로드는 일시적으로 사용하는 것이기 때문에 프린터를 기동시킬 때마다 다시 다운로드해야 하는 단점이 있다. 하지만 원하는 캐릭터를 인자하기 위해서는 이러한 불편은 감수해야 한다.

●다운로드의 실행예

그럼 이제 실질적으로 다운로드를 하여 보자. 다운로드에

〔그림 1〕 사용자 정의 패턴의 예와 잘못된 경우



(a) Draft문자 디자인을 위한 눈금 (b) 정의된 문자 (c) 잘못된 경우

※ 도트를 눈금에 맞출때 수평선으로는 움직일 수 없고 도트끼리 중복되지 않는한 수직선으로는 움직일 수 있다.

관련된 제어코드는 표 1에 나타낸 바와 같으며 다운로드를 하려면 먼저 사용자 정의문자의 패턴을 설계해야 한다. 그런 다음에 이 패턴을 프린터의 램에 복사한다. 그 이후부터는 다운로드를 할 수 있다. 다운로드를 실행하려면 표 1에서 보인 바와 같이 제어코드를 프린터에 전송하고 일반캐릭터를 사용하는 것과 같이 사용하면 된다.

그러면 드래프트 모드에서 다운로드하는 방법을 알아 보자. 먼저 사용자 문자를 정의한다. 문자를 정의하는 것은 그림 1에서 설명하는 것과 같다. 그림 1의 (a)와 같이 대부분의 문자들은 밑의 2컬럼을 다음 캐릭터와 붙지 않도록 공간을 두기 때문에 실질적으로 사용되는 도트는 가로 9도트 세로 7도트(8도트 모두 사용해도 됨)인 셈이다.

패턴을 설계할 때 주의할 것은 중복된 도트나 수평선 위에 도트를 그려서는 안된다는 점이다. 만약 중복된 도트가 있다면 프린터는 한 도트 만을 인쇄할 것이다.

그림 1과 같이 정의했으면 베이직 프로그램으로 간단하게 실행하여 보자. 다운로드를 사용하려면 먼저 캐릭터를 ROM에서 RAM으로 복사를 한다.(제어코드 ESC":" 사용) 그런다음 사용자 정의문자를 선택한다.(ESC "%"사용) 마지막으로 ESC "&"를 사용하여 패턴을 정의하면 된다. 잘 이해가 가지 않은다면 리스트 1을 실행하여 보면 쉽게 알 수 있다. 리스트 1은 드래프트 모드에서의 다운로드 프로그램을 보인 것이다.

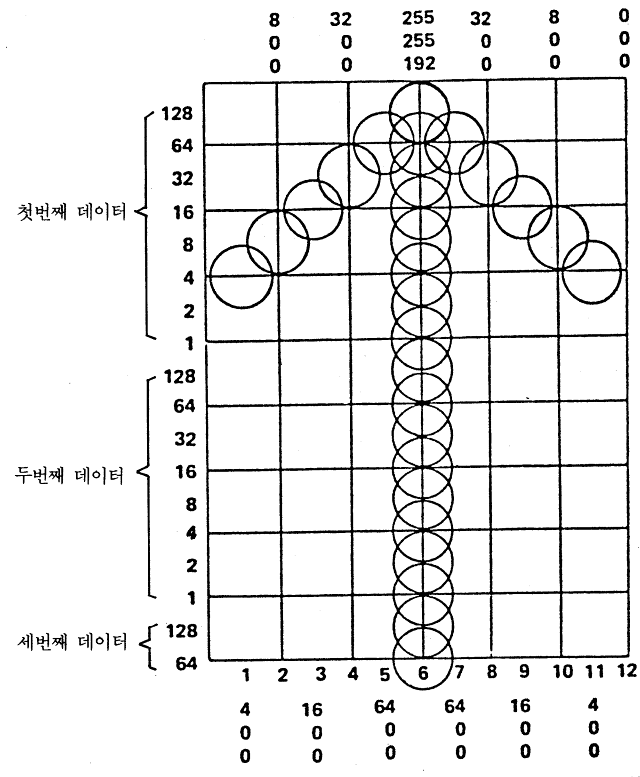
NLQ모드에서 다운로드하는 것은 좀더 복잡하다. 문자체가 정밀한 만큼 사용하기도 어려운 것은 당연하다. 데이터 전송은 그림 2와 같다. 먼저 위의 4비트를, 다음은 두번째 4비트를 그리고 아래 2비트를 보내면 된다. 나머지는 앞의 정의와 동일하지만 정의하기 전에는 NLQ 모드로 전환되어 있어야 한다.

리스트 2는 NLQ에서의 다운로드하는 리스트 예를 보인 것이다.

〔표 1〕 다운로드에 사용되는 프린터 제어코드

ESC &	〔형식〕 ESC "&" NUL d_1 d_2 … d_n 이 명령은 현재 선택된 모드를 재정의 한다. d_1과 d_2는 58에서 63(3AH에서 3FH)까지 가능함.
ESC :	〔형식〕 ESC : NUL NUL NUL LX-800 ROM문자 세트를 RAM에 복사한다.
ESC %	〔형식〕 ESC%n 만약 n이 1이면 사용자정의 문자가 되며 n이 0이면 일반 문자가 세트된다.

〔그림 2〕 NLQ문자의 패턴과 전송



●한글과의 문제

한글은 모듈에 의해서 사용되어 지는 것이므로 일반적인 다운로드는 사용할 수 없다. 그러므로 모드를 수시로 바꾸어 주어야하는 번거로움이 있다.

딥스위치와 제어패널의 조작

프린터의 뒤를 보면 딥 스위치라는 작은 스위치가 있다. 이것은 프린터의 인자 방식을 변환하여주는 스위치이다. 보통 회사에서 나올 때에는 모두 OFF로 되어 있는데 일반적으로 이 스위치를 사용하는 사람은 별로 없는 것같다. 딥스위치의 기능은 표 2와 같다. 여기서 설명할 것은 2번과 3번 스위치이다. 2번 스위치는 숫자 0을 프린트할 경우 0자 안에 슬래쉬(/)를 할 것인가 안할 것인가를 선택한다. 일반적으로 알파벳0과 혼돈이 되므로 2번 스위치를 ON하여 두 글씨를 구분하는 것이 좋다.

3번 스위치는 128번 캐릭터부터 255번 캐릭터까지 해당하는 문자의 모드를 변환하여 준다. 독자중에서는 화면상에 있는 격자를 프린트 스크린(〈Shift〉와 〈PrtSc〉를 동시에 누름)할 때 프린팅이 깨져서 (사실은 이탤릭체) 출력되는 경험을 했을 것이다. 이탤릭체를 사용하는 경우는 드물기 때문에 이 스위치도 ON하여 사용하면 편리하다. (화면 1은 딥스위치를 조절한 후의 128번부터 255번까지의 문자세트를 나타낸 것이다.)

필자의 경우는 80컬럼의 프린트 용지보다 낱장용지를 애용하는 편이다. 왜냐하면 프린트 트랙터에 걸어야하는 용지의 낭비를 없애기 위해서이다. LX-800프린터는 트랙터가 위에 부착되어 있어 용지를 끌어 당기게 되어 있다. 그러면 한 페이지를 인쇄한 후에 그것을 뜯어내면 다음에는 한 페이지를 희생하여야하는 단점이 있다. 그러나 낱장용지는 한 장을 가지고도 사용이 가능하며 뒷장도 사용이 가능하다.

LX-800프린터에서는 AUTO LOAD기능이 있어 잘 사용하고 있다. 독자들에게도 한번쯤은 이 기능을 사용하여 볼 것을 권유하고 싶다. 용지는 습식 복사용지인 A4용지이면 적당하다.

이 경우 딥스위치(두번째 딥스위치)의 1번을 ON 상태로 하여 사용하여야 한다.

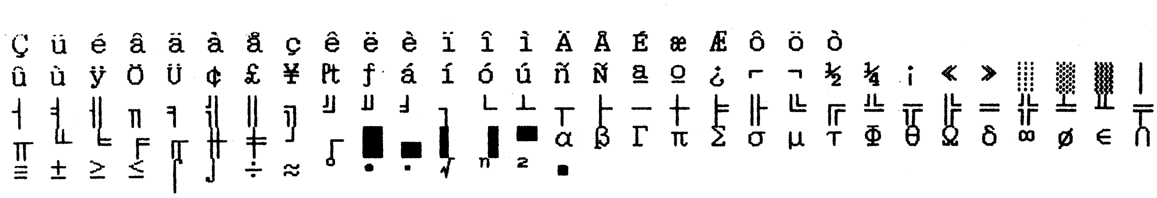
〔리스트 1〕 Draft상태에서의 다운로드

```
100 FOR P=58 TO 63: LPRINT CHR$(P);" ";: NEXT P
110 LPRINT
120 K=1: IF K>3 THEN A=58 ELSE A=60
125 REM ROM에서 RAM으로 복사한다
130 LPRINT CHR$(27);":"CHR$(0)CHR$(0)CHR$(0);
135 REM 사용자 정의 문자로 초기화한다
140 LPRINT CHR$(27)"%"CHR$(1)CHR$(0);
145 REM 사용자 정의문자의 패턴을 정의한다
150 LPRINT CHR$(27)"&"CHR$(0)CHR$(A)CHR$(A-1+K);
160 FOR Z=1 TO K
170 LPRINT CHR$(128);
180 FOR X=1 TO 9
190 READ R
200 LPRINT CHR$(R);: NEXT X
210 LPRINT CHR$(0)CHR$(0);
220 NEXT Z
230 DATA 0,80,170,0,47,0,170,80,0
240 FOR P=58 TO 63: LPRINT CHR$(P);" ";: NEXT P
```

〔리스트 2〕 NLQ모드에서의 다운로드

```
100 LPRINT CHR$(27);"x";CHR$(1); 'NLQ 모드로 전환한다
110 LPRINT CHR$(27);":";CHR$(0);CHR$(0);CHR$(0);'ROM을RAM에복사한다
120 LPRINT CHR$(27);"%";CHR$(1);CHR$(0); '사용자 문자로 정의
130 READ A$
140 IF A$="A" THEN END
150 LPRINT CHR$(27);"&";CHR$(0);A$;A$;
155 LPRINT CHR$(0);CHR$(12);CHR$(0);
160 FOR K=0 TO 36
170 READ F : LPRINT CHR$(F);
180 NEXT K
190 REM 문자 출력
200 LPRINT CHR$(27);"%";CHR$(0);CHR$(0);      '일반 문자로 정의
210 LPRINT A$; " becomes ";
220 LPRINT CHR$(27);"%";CHR$(1);CHR$(0);      '사용자 문자로 정의
230 LPRINT A$
240 GOTO 130
250 DATA "<",4,0,0,8,0,0,16,0,0,32,0,0
260 DATA 64,0,0,255,255,192,64,0,0,32,0,0
270 DATA 16,0,0,8,0,0,4,0,0,0,0,0,
280 DATA "A"
```

〔화면 1〕 3번 딥스위치 ON상태



LX-800 프린터의 콘트롤 패널(Control Pannel)은 라인피드(Line Feed)나 폼피드(Form Feed)의 기능만 있는 것은 아니다. 프린터가 지원하는 폰트는 5가지 정도가 있는데 이 모든 것을 콘트롤 패널에서 변환할 수 있다.

DTRAFT와 NLQ는 ON LINE상태에서 변환한다는 것은 누구나 알고 있지만 다른 폰트(Emphasized, Condensed, Elite) 를 변환하는 방법을 아는 사람은 드물다.

이 방법은 다음과 같은 순서로 하면 된다.

1. On Line 스위치와 Form Feed 스위치를 동시에 누른다. 그러면 Beep음이 한번 나면서 On Line의 램프가 깜빡이게 된다. 만약 이 작업이 실패하면 Beep음이 두번 나고 NLQ모드로 들어가 버린다.
2. 불이 깜박이면 On Line 스위치를 원하는 모드의 번호만큼 누른다. 예를 들면 Elite모드로 전환하고 싶으면 On Line 스위치를 4번 누른다.
3. 다음은 Form Feed 스위치를 누르고 Line Feed 스위치를 누른후 다시 On Line 스위치를 누른다.

그러면 여러가지 폰트를 간단한 조작으로 얻을 수 있다. 위의 방법으로 축소문자나 엘리트(Elite)문자 등을 사용할 수 있는데 이것은 중복되어도 무관하다. 명령이 중복되면 인쇄 효과도 중복되어 나타난다. 즉 엘리트체에 축소모드 그리고 NLQ의 형식으로 출력 가능하다(그림 3 참조).

유용한 유틸리티들

LX-800프린터로 하드 카피를 해본 사람이라면 누구나 쉽게 줄이 생기는 것을 볼 수 있다. 여담이지만 필자는 이것 때문에 LX-800을 구입하지 않으려고까지 하였다. 하지만 가격이 저렴한 관계로 또한 기능이 뛰어난 관계로 여러모로 생각하여 결국은 구입을 하고 말았다. 왜냐하면 이러한 줄생김은 간단한 램상주 프로그램으로 제어할 수 있기 때문이다.

하드카피시 줄이 생기는 버그가 어떠한 이유에서 생기는지는 모르겠지만 어쨋든 결과가 중요할 것이다. 보통 이미지 프린팅을 할 경우 18 / 216 인치로 라인 피드를 하고 있다. (제어 코드 ESC "3"이나 ESC "J" 혹은 ESC "A") 이러한 이유로 프린트시 간격이 벌어지는 것을 17 / 216 인치로 조정하면 될 것이다.

이 방법에는 두가지가 있다. 일반적으로 많이 사용하는 허큘리스 그래픽 모드에서 Graph-X라는 패키지 안의 HARD-COPY. COM을 사용하면 간단히 해결된다. 그러나 다른 패키지(뉴스마스터나 닥터할로우)에서는 패키지가 프린터를 직접 제어하므로 램상주 프로그램으로 인터럽트 17번(프린트 제어)을 가로채야 한다.

●HARDCOPY.COM을 이용

Graph-X라고 하는 허클리스용 그래픽 패키지가 있다. 이것은 인터럽트 10h를 제어하여 허클리스 보드를 사용하는

〔표 2〕 딥 스위치 기능

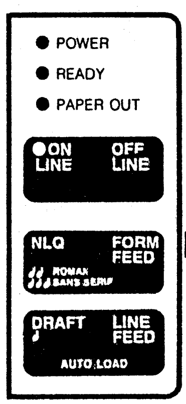
스위치 번호	기 능	ON상태	OFF상태
1-1	축소모드와 정상모드를 선택	축소문자	정상문자
1-2	제로의 출력형태 선택	Ø	0
1-3	캐릭터 테이블 선택	Graphics	Italics
1-4	Paper-out 감지상태	취 소	활 성 화
1-5	프린트 quality 선택	NLQ	Draft
1-6 1-7 1-8	각국의 문자세트 선택		

(a) 딥스위치 그룹 1

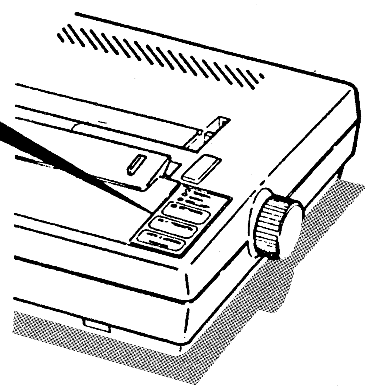
스위치 번호	기 능	ON상태	OFF상태
2-1	페이지 길이 선택	12인치	11인치
2-2	컷 시트 피더 모드	선택	선택하지 않음
2-3	절단선 스킵	1인치	0
2-4	캐리지 리턴후의 라인 피드	CR+LF	CR만

(b) 딥스위치 그룹2

〔그림 3〕 콘트롤 패널과 TYPE모드 선택



(a) 콘트롤 패널



(b) TYPE모드 선택

모드	인자상태(기능)
1	Emphasized : Emphasized & NLQ (ROMAN) Emphasized & NLQ (ROMAN)
2	Double-Strike : Double-Strike & NLQ(ROMAN) Double-Strike & NLQ(ROMAN)
3	Condensed : Condensed & Draft Condensed & Draft
4	Elite : Elite & NLQ (ROMAN) Elite & NLQ (ROMAN)

사람들을 위한 기능을 더 추가하여 사용하는 패키지이다. 램상주 프로그램이므로 어떠한 프로그램 언어에서도 그래픽이 가능하며(심지어는 포트란까지) 물론 그 안에는 허큘리스의 그래픽 모드(720×348)를 프린트하는 프로그램도 들어 있다. 이것이 바로 HARDCOPY. COM이다.

이것은 램상주로 있으면서 〈Shift〉+〈PrtSc〉의 키가 눌려지면 호출이 된다. 이 키가 눌려지면 컴퓨터 내부에서는 인터럽트 5h번이 호출되며 HARDCOPY.COM은 이 인터럽트를 수정하게 된다. HARDCOPY.COM을 수정하려면 먼저 여기에 사용되는 프린트 제어명령을 수정하여야 한다.

패키지 안에 있는 화일을 보면 PRINTER.DEF라고 하는 프린트 정의화일이 있는데 이것을 LX800.COM이라는 화일로 다시 카피를 한다. 그런 다음 에디터를 이용하여 **화면 2**와 같이 수정을 한다. HARDCOPY.COM을 실행할 때에는

```
A>HARDCOPY LX800.DEF
```

와 같은 방법으로 정의화일을 파라미터로 주면 된다.

프린터를 하려면 〈Shift〉+〈PrtSc〉를 누른다음 그래픽모드에 따라 1번에서 6번까지를 선택하면 된다(표 3 참조).

●램상주 프로그램의 이용

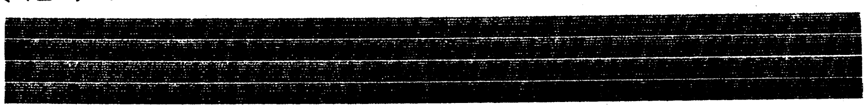
만약 다른 패키지에서 사용을 하여야 한다면 램상주 프로그램을 이용하여야 할 것이다. **리스트 3**은 매크로 어셈블러를 이용하여 작성한 변환 프로그램이다. 이 프로그램은 패키지를 사용하기 전에 실행하기만 하면 된다. 필자가 NEWS MASTER Ⅱ와 Dr. Harlo Ⅲ에서 사용하여 본 결과 정말 깨끗한 인쇄물을 얻을 수 있었다(어셈블리환경은 갖추지 못한 경우는 **리스트 4**의 베이직 프로그램을 입력해서 사용하기 바란다).

프로그램은 프린터제어 인터럽트로 들어 오는 데이터를 체크하여 18 / 216 인치로 라인피딩하라는 명령이 들어오면 18의 데이터를 17로 변환하여 준다. 물론 일반 캐릭터는 정상적으로 프린트된다(화면 3 참조).

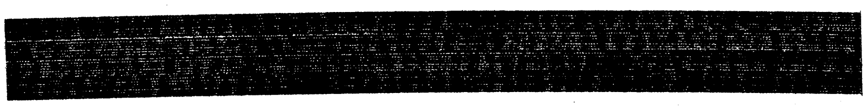
〔화면 2〕 LX800. DEF의 수정

```
KEYWORDS MUST BE FOLLOWED BY AN EQUAL SIGN FOLLOWED BY ESCAPE SEQUENCE
CONTROL SEQUENCES ARE TERMINATED BY A \41
KEYWORDS MAY OCCUR IN ANY ORDER;ONLY THE LAST OCCURENCE WILL BE RECORDED
AN EQUAL SIGN SHOULD NOT OCCUR IN THE ESCAPE SEQUENCES
NON CHARACTERS ARE INDICATED BY \ FOLLOWED BY HEX VALUE
                IBM PRINTER
width           =8
topbit          =7
setup           =\1B@\1b3\17\41 이 부분을 /1B@/1B3/17/41로 바꾼다.
header          =\1bL\D0\02\41
tail            =\0d\0a\41
restore         =\1b@\40\41
normal          =1
normal alt      =2
reverse         =3
reverse alt     =4
normal both     =5
reverse both    =6
```

〔화면 3〕 하드카피 교정 전과 후 비교



(a) 전



(b) 후

〔표 3〕 HARDCOPY.COM의 키보드조작

입 력 키	기 능
〈SHIFT〉+〈PrtSc〉-1	그래픽 페이지 0를 인쇄
〈SHIFT〉+〈PrtSc〉-2	그래픽 페이지 1을 인쇄
〈SHIFT〉+〈PrtSc〉-3	페이지 0를 역상으로 인쇄
〈SHIFT〉+〈PrtSc〉-4	페이지 1을 역상으로 인쇄
〈SHIFT〉+〈PrtSc〉-5	두 페이지를 모두 인쇄
〈SHIFT〉+〈PrtSc〉-6	두 페이지를 모두 역상으로 인쇄
〈SHIFT〉+〈PrtSc〉-Space Bar	정상적인 텍스트 모드를 인쇄

＊참고 : 여기서 ＋는 동시에 누르는 것이며, －는 앞의 것을 누른 후에 누른다는 뜻임.

맺음말

옛날에는 컴퓨터를 구입했다 하면 본체와 모니터 그리고 디스크 드라이브를 말하였지만 지금은 프린터에 하드디스크까지 장만해야만 한 세트를

〔리스트 3〕 매크로 어셈블러를 이용한 변환프로그램

```
;========================================================
;
;    LX800.ASM   CONVERT PROGRAM for LX-800 PRINTER
;                  by Shim, Seongho
;
;         MYONG JI UNIV. ELECTRONICS ENGINEERING
;                           &
;                     P. R. D. A.
;
;               Created : 1989 . 2 . 7
;
;========================================================
;
CSEG        SEGMENT
            ASSUME  CS:CSEG
            ORG     100h
START:
            CALL    INITIALIZE          ; 인터럽트 벡터를 변경한다.
            MOV     DX,OFFSET END_OF_RAM
            INC     DX
            INT     27h                 ; 램상주시킨다.
;
NEW_VECTOR  PROC    NEAR
;
            CMP     AH,00       ; AH 와 0을 비교한다.
            JZ      N1          ; 만약 0이면 N1으로 분기한다.
            MOV     DX,00h      ; 문자 출력이 아니면 원래 인터럽트를
                                  호출한다.
            INT     64h         ; 원래 INT 17h을 호출한다.
            IRET
;
N1:
            PUSH    DS          ; 사용될 레지스터를 대피시킨다.
            PUSH    BX
            PUSH    DX
            MOV     BX,CS       ; CS 레지스터의 값을
            MOV     DS,BX       ; DS 에 복사한다.
            CMP     AL,1Bh      ; AL이 제어 코드인지를 확인한다.
            JZ      N2          ; 만약 제어 코드이면 N2로 분기한다.
            CMP     AL,33h      ; ESC '3' 인지를 확인한다.
            JZ      N3          ; 만약 같다면 N3으로 분기한다.
            CMP     AL,4Ah      ; ESC 'J' 인지를 확인한다.
            JZ      N3          ; 만약 같다면 N3으로 분기한다.
            CMP     AL,18h      ; n/216 의 n이 18h인가를 확인한다.
            JZ      N4          ; 만약 같다면 N4로 분기한다.
;
            MOV     BX,OFFSET FLAG
            MOV     BYTE PTR [BX],00h
            JMP     N4_2
N2:
            MOV     BX,OFFSET FLAG
            MOV     BYTE PTR [BX],01h
            JMP     N4_2
N3:
            MOV     BX,OFFSET FLAG
            CMP     BYTE PTR [BX],01h
            JZ      N3_1
            MOV     BYTE PTR [BX],02h
            JMP     N4_2
N3_1:
            MOV     BYTE PTR [BX],02h
            JMP     N4_2
N4:
            MOV     BX,OFFSET FLAG
            CMP     BYTE PTR [BX],02h
            JZ      N4_1
            MOV     BYTE PTR [BX],00h
            JMP     N4_2
N4_1:
            MOV     AL,17h              ; AL이 18h인 것을 17h로
N4_2:                                     변환한다.
            MOV     DX,00h
            INT     64h
            POP     DX                  ; 보관된 레지스터들을 돌려준다.
            POP     BX
            POP     DS
            IRET                        ; RETURN
;
NEW_VECTOR  ENDP
;
INITIALIZE  PROC    NEAR
            MOV     AX,3517h
            INT     21h             ; 프린트 벡터를 읽어들인다.
            MOV     DX,ES           ; 읽어진 세그먼트 값을
            MOV     DS,DX           ; DS에 복사한다.
            MOV     DX,BX           ; 읽혀진 오프셋을 DX에 복사한다.
            MOV     AX,2564h        ; 프린트 벡터를 64h에 대피시킨다.
            INT     21h             ; DOS 기능 호출
            MOV     DX,CS
            MOV     DS,DX                   ; CS를 DX에 복사한다.
            MOV     DX,OFFSET NEW_VECTOR    ; 새 벡터의 주소를 DX에
            MOV     AX,2517h                ;            저장한다.
            INT     21h                     ; 새로운 벡터를 지정한다.
            RET
INITIALIZE  ENDP
;
  FLAG      DB  00
;
END_OF_RAM:
;
CSEG        ENDS
            END     START
```

〔리스트 4〕

```
100 '
110 'LX800.COM Maker
120 'by Shim Seong-ho  on 89/02/13
130 '
140 DEFINT A-Z : KEY OFF : CLS
145 PRINT "Welcome LX800.COM Maker " : PRINT
150 INPUT "Input Path :",PATH$
160 OPEN PATH$+"LX800.COM" AS #1 LEN = 1
170 FIELD #1,1 AS A$
180 READ B
190 IF B = 999 THEN 230
200 LSET A$=MKI$(B)
210 PUT #1
220 GOTO 180
230 PRINT : PRINT "LX800.COM has been created."
235 CLOSE #1
240 END
250 DATA 232, 103,   0, 186, 136,   1,  66, 205,  39, 128
260 DATA 252,   0, 116,   6, 186,   0,   0, 205, 100, 207
270 DATA  30,  83,  82, 140, 203, 142, 219,  60,  27, 116
280 DATA  21,  60,  51, 116,  26,  60,  74, 116,  22,  60
290 DATA  24, 116,  38, 187, 135,   1, 198,   7,   0, 235
300 DATA  46, 144, 187, 135,   1, 198,   7,   1, 235,  37
310 DATA 144, 187, 135,   1, 128,  63,   1, 116,   6, 198
320 DATA   7,   2, 235,  23, 144, 198,   7,   2, 235,  17
330 DATA 144, 187, 135,   1, 128,  63,   2, 116,   6, 198
340 DATA   7,   0, 235,   3, 144, 176,  23, 186,   0,   0
350 DATA 205, 100,  90,  91,  31, 207, 184,  23,  53, 205
360 DATA  33, 140, 194, 142, 218, 139, 211, 184, 100,  37
370 DATA 205,  33, 140, 202, 142, 218, 186,   9,   1, 184
380 DATA  23,  37, 205,  33, 195,   0, 999
```

갖추었다고 한다 . 가격이 그만큼 내려간 이유도 있겠지만 컴퓨터는 혼자서만은 무용지물이기 때문이다. 그렇기 때문에 프린터라는 것은 주변기기로서 정말 필요한 존재이며 그 기능들을 잘 알아야 하는 것은 당연한 일이다.

확장 함수키의 정의

IBM의 베이직프로그램에서 함수키(Function Key)를 재정의(Redefined)해 본 사람이라면 누구나 프로그램이 끝났을 때 이 함수키를 원래 정의대로 복구시킬 수 있는 명령어가 없다는 점에 대해 한번쯤 궁금하게 여겨 본 경험이 있을 것이다.

대부분 프로그램이 끝났을 때 함수키를 원점으로 복구시키는 방법에는 그 함수키의 디폴트(Default) 값에 다시 그 함수키를 재정의하거나 디스에이블(Disable)시켜 종료하는 것이 있다.

그러나 여기에서는 그와 다른 방법으로 프로그램이 끝났을 때 함수키를 원점으로 복구시키는 것을 소개하고자 한다. 지금 소개하고자 하는 방법의 열쇠는 소프트키 버퍼(Soft Key Buffer)라고 부르는 것에 있다. 만일 독자들이 IBM PC/XT를 갖고 있다면 이 소프트키 버퍼를 쉽게 찾을 수 있다. 이 버퍼를 찾는 소프트웨어는 메모리 세그먼트내의 1619 메모리번지(Location)에서 시작한다.

그러나 IBM 호환기종일 경우에는 반드시 이 메모리번지가 1619는 아니기 때문에 호환기종을 사용할 경우에는 리스트 1의 프로그램을 실행시킨다. 그러면 그 기종에서의 소프트키 버퍼의 메모리번지를 찾을 수 있다.

그럼 독자들은 갖고 있는 호환기종에서 이 리스트 1의 프로그램을 실행시켜 소프트키 버퍼의 메모리번지를 찾고, 그 다음에는 이 소프트키 버퍼를 실행시키기 전에 IBM의 프로그램에 있는 1619 소프트키 버퍼의 어드레스를 호환기종내의 프로그램에 있는 소프트키 버퍼의 메모리번지로 바꿔주어야 한다. 이 번지가 위치할 수 있는 라인은 각 베이직 프로그램내의 REM문으로 표시된다.

함수키의 정의와 저장

소프트키 버퍼는 함수키에 대한 정의를 저장한 메모리의 일부분이다. 이 소프트키 버퍼에 저장되어 있는 하나의 함수키에 다른 하나의 기능을 정의할 경우, 그 정의는 바뀐다. 키를 정의하는데 사용되는 캐릭터는 최대 15자리까지 가능하다. 그러나 만일 버퍼의 메모리번지를 PEEK로 지정했을 경우 각 키를 정의하는 캐릭터의 영역이 최대 15가 아니라 16자리까지 지정이 가능하다는 것을 알게 될 것이다.

한편 각 함수키가 할당된 번지의 번호를 알면 버퍼의 내용을 쉽게 저장할 수 있으므로 함수키의 정의 내용을 보존하기 쉽다.

리스트 2의 프로그램은 소프트키 버퍼의 내용을 배열 형태로 읽어 들이면서 함수키의 정의를 정하는 일을 수행한다. 따라서 리스트 2의 프로그램을 실행하면 각 함수키에 새로운 기능을 지정할 수 있다. 결국 리스트 1의 프로그램은 프로그램의 배열내용을 다시 소프트키 버퍼로 POKE시켜 함수키의 원래 기능을 복구시키도록 해준다.

아직도 함수키를 정의하는데 있어서 15자리까지로 한정되어 있는데도 불구하고 버퍼내에서 각 함수키의 정의가 왜 최대 16자리까지 지정이 가능한가를 의심하는 독자들이 있다면 이 함수키들을 디스에이블시켜 보면 그 해답을 얻을 수 있을 것이다.

예를 들어 F1에 대해 저장된 16자리를 PEEK로 지정하고

〔리스트 1〕 호환기종에서 소프트키 버퍼를 찾아주는 프로그램

```
100 DEF SEG: SCREEN 0:WIDTH 80:X=0
110 CLS:PRINT "MEMORY LOCATION ";:LOCATE ,20
120 KEY 1,"LIST":A=ASC("L")
130 IF PEEK(X)=A THEN GOSUB 150 ELSE PRINT X;:LOCATE 1,20
140 X=X+1:GOTO 130
150 IF CHR$(PEEK (X+1))<>"I" THEN RETURN
160 IF CHR$(PEEK (X+2))<>"S" THEN RETURN
170 IF CHR$(PEEK (X+3))<>"T" THEN RETURN
180 CLS:PRINT "MEMORY LOCATION ";X
190 FOR J=1 TO 10:PRINT "F";J;:FOR K=0 TO 15
200 IF PEEK(X+16*(J - 1)+K)>0 THEN PRINT CHR$(PEEK(X+16*(J - 1)+K));ELSE 220
210 NEXT K
220 PRINT:NEXT J
230 BEEP:INPUT "IS THIS IT ";Q$
240 IF Q$="Y" OR Q$="y" THEN END ELSE X=X+1:CLS:GOTO 110
```

(원래는 LIST로 정의됨) ASCII값을 프린트하면 그 내용은 다음과 같다.

LIST0000000000000

그러나 F1을 디스에이블시키면 소프트키 버퍼는 다음과 같이 표현된다.

0IST0000000000000

이 예에서 베이직은 함수키 정의의 끝을 0으로 표시한다는 것을 알 수 있다. 리스트 3의 프로그램을 실행시켜 보면 이 내용이 입증될 것이다.

즉, 리스트 3의 프로그램은 단지 함수키 정의의 첫번째 캐릭터를 저장하고 이들을 디스에이블함으로서 함수키를 복구할 수 있다는 것을 입증한다. (물론 그 함수키는 정의의 첫번째 캐릭터만을 바꿈으로서 디스에이블된다는 가정하에) 따라서 리스트 3의 프로그램을 함수키 복구에 사용할 경우에는 리스트 2에서 160바이트를 저장하는데 비해 10바이트만 정의해도 된다.

확장 정의

각 함수키 정의의 첫번째 캐릭터만을 저장하고, 이를 디스에이블함으로서 함수키를 복구할 수 있다는 사실을 안 것은 꽤 흥미로운 일이다. 그러나 리스트 2와 3에서 저장된 바이트가 10과 160바이트로 차이가 난다는 것이 왠지 석연치 않은 느낌은 줄 것이다.

여기서 알 수 있는 것은 버퍼내 함수키의 16번째 자리를 변경시킴으로써 함수키를 정의하는데 사용되는 캐릭터 영역을 확장시킬 수 있다는 것이다. 즉, 이 캐릭터의 영역을 확장한다는 것은 함수키를 정의하는데 있어서 더 많은 캐릭터로 지정할 수 있다는 말이 된다. (하지만 다음 F9함수키를 사용한다는 전제하에서)

예를 들어 SCREEN 0,0,0과 WIDTH 80에서 프로그램을 편집 할 경우 이들이 비록 15캐릭터를 초과한다할지라도 리스트 4의 프로그램을 사용하면 F9 함수키를 세트할 수 있다.

〔리스트 2〕 함수키의 정의

```
90 REM LINES WHICH USE 1619 OFFSET ARE 140 AND 250
100 SCREEN 0:WIDTH 80:CLS:DEF SEG:OPTION BASE 1
110 KEY ON:DIM K$(10):FOR X=1 TO 10:K$(X)=STRING$(16,0):NEXT X:'STORAGE AREA FOR
 FUNCTION KEYS
120 REM SAVE FUNCTION KEYS
130 FOR X=1 TO 10:FOR J=0 TO 15
140 MID$(K$(X),J+1,1)=CHR$(PEEK(1619+16*(X-1)+J))
150 NEXT J,X
160 REM REDEFINE FUNCTION KEYS WITH LETTERS(THIS IS ONLY AN EXAMPLE)
170 FOR X=1 TO 10:KEY X,CHR$(X+64):NEXT X:KEY LIST
180 PRINT "FUNCTION KEYS ARE REDEFINED":PRINT "PRESS ANY KEY TO RESTORE"
190 KB$=INKEY$:IF KB$="" THEN 190
200 REM RESTORE FUNCTION KEYS
210 FOR X=1 TO 10
220 KEY X,K$(X)
230 NEXT X:CLS
240 FOR X=1 TO 10
250 J=ASC(MID$(K$(X),16,1)):IF J>0 THEN POKE 1619+16*(X-1)+15,J
260 NEXT X:CLS
270 KEY LIST
```

〔리스트 3〕 함수키의 복구

```
90 REM LINES WHICH USE 1619 OFFSET ARE 140 AND 220
100 SCREEN 0:WIDTH 80:CLS:DEF SEG
110 KEY ON:K$=STRING$(10,0):'STORAGE AREA FOR FUNCTION KEYS
120 REM SAVE FUNCTION KEYS
130 FOR X=1 TO 10
140 MID$(K$,X,1)=CHR$(PEEK(1619+16*(X-1)))
150 NEXT X
160 REM DISABLE FUNCTION KEYS
170 FOR X=1 TO 10:KEY X,"":NEXT X:KEY LIST
180 PRINT "FUNCTION KEYS ARE DISABLED":PRINT "PRESS ANY KEY TO RESTORE"
190 KB$=INKEY$:IF KB$="" THEN 190
200 REM RESTORE FUNCTION KEYS
210 FOR X=1 TO 10
220 POKE 1619+16*(X-1),ASC(MID$(K$,X,1))
230 NEXT X:CLS
240 KEY LIST
```

〔리스트 4〕 함수키의 확장정의

```
90 REM LINES WHICH USE THE 1619 OFFSET ARE 180,290,390,440,470
100 DEF SEG:STK$=STRING$(128,0):SCR$=STRING$(37,0):RESTORE 110:FOR X=1 TO 37:REA
D J:MID$(SCR$,X,1)=CHR$(J):NEXT X:SCR!=PEEK(VARPTR(SCR$)+1)+256*PEEK(VARPTR(SCR$
)+2)
110 DATA 85,137,229,139,118,6,41,192,138,4,139,116,1
120 DATA 1,240,137,196,184,0,6,187,0,7,185,0,2
130 DATA 186,80,24,85,205,16,92,93,202,2,0
```

```
140 SCREEN 0:WIDTH 80:CLS
150 T$="FUNCTION KEY DEFINITION"
160 LOCATE 2, (40-.5*LEN(T$)):PRINT T$
170 PRINT:PRINT
180 X=1:J=1:K=1619
190 K$=STRING$(160,0):KN$=STRING$(160,0):K=K-1
200 L=PEEK(J+K)
210 WHILE L<>0
220 MID$(K$,J,1)=CHR$(L)
230 J=J+1:L=PEEK(J+K)
240 WEND
250 PRINT "FUNCTION KEY ";X;": ";MID$(K$,1,J-1)
260 PRINT:PRINT "ENTER NEW DEFINITION OR PRESS ENTER TO LEAVE UNCHANGED"
270 LINE INPUT Q$:IF LEN(Q$)>0 THEN GOSUB 300:IF ER=1 THEN ER=0:GOTO 250
280 IF X+FIX(J/16)>9 THEN GOTO  380
290 X=X+1+FIX(J/16):K=1619+16*(X-1)-1:J=1:CALL SCR!(STK$):LOCATE 5,1:GOTO 200
300 INPUT "Do you want a carriage return (Y/N)";Q1$
310 IF Q1$="Y" OR Q1$="y" THEN Q$=Q$+CHR$(13)
320 IF LEN(Q$)<16 THEN J=LEN(Q$):KEY X,Q$:RETURN
330 M=1:N=16*(X-1)+1:IF N+LEN(Q$)>160 THEN BEEP:PRINT "Too long":ER=1:RETURN
340 MID$(KN$,N,1)=MID$(Q$,M,1)
350 M=M+1:N=1+N:IF M<=LEN(Q$) THEN 340
360 IF LEN(Q$) > J THEN J=LEN(Q$)
370 RETURN
380 FOR X = 1 TO 10
390 IF ASC(MID$(KN$,16*(X-1)+1,1)) > 0 THEN FOR J = 16*(X-1)+1 TO 16*X:POKE 1619
+J-1,ASC(MID$(KN$,J,1)):NEXT J
400 NEXT X:CLS:KEY LIST
410 KB$=INKEY$:IF KB$="" THEN 420 ELSE 410
420 PRINT:INPUT "Do you want to save function keys  as a BLOADable file (Y/N)";Q$
430 IF Q$="Y" OR Q$="y" THEN INPUT "Filename";F$ ELSE END
440 BSAVE F$,1619,159:PRINT
450 PRINT "To load your function key file, use these commands:"
460 PRINT:PRINT
470 PRINT "DEF SEG:BLOAD ";CHR$(34);F$;CHR$(34);",1619:CLS":END
```

여기서 소프트키 버퍼의 사이즈 즉 버퍼내의 F9를 정의한 캐릭터 수를 증가시키지 않았으므로 F10은 사용할 수 없게 된다.

또한 **리스트 4**의 프로그램은 새로운 함수키의 정의를 다른 프로그램으로부터 BLOAD될 수 있도록 화일로서 저장할 수 있도록 한다. 만약 이를 시도할 경우 화일을 BLOAD할 때 버퍼 어드레스인 1619를 생략시키지 말아야 한다는 점을 명심해야 한다. 왜냐하면 베이직의 세그먼트가 원래 만들어진 화일과 똑같은 화일이 된다는 보장이 없기 때문이다.

〈용어해설〉

●함수키(Function Key)
타이프라이터나 단말기의 키보드상에서 프린트 이외의 제어(Control) 목적에 이용되는 키이다.

●디폴트(Default) 값
유저가 새로운 값을 지정하지 않았을 때 시스템은 자동적으로 먼저번에 최종적으로 사용된 값을 선택하게 되는데 이 값을 디폴트 값이라고 한다.

●복근(Restore)
레지스터, 카운터, 메모리 어드레스 등을 이전의 값 또는 상태로 세트하거나 세이브해둔 것을 원래대로 복귀시키는 것이다.

●함수(Function)
특정한 목적 또는 정의된 동작을 말한다. 하드웨어의 특성에 기초를 둔 작동을 말할 때는 기능이라고 표현하며 소프트웨어에서는 프로그램의 목적에서부터 프로그램안에서 특정한 일을 행하는 경로 등을 가리키는 의미로 함수라고 표현한다.

●휨웨어(Firmware)
칩내부에서 조직된 프로그램 휨웨어는 불휘발성으로 전원이 끊어져도 변하지 않는다. 모니터 등에 주로 사용되는 프로그램은 휨웨어로 되어 있다.

●GIGO
두가지 경우에 이를 약자로 쓰는데, Gabage In, Gabage Out이란 약자를 대신할 때는 틀린 자료를 입력하면 반드시 틀린 답만이 나온다는 의미이고, Gabage In, Gospel Out이란 약자를 대신할 때는 틀린 자료를 입력해도 바른 답이 나온다는 의미이다.

●캐릭터(Character)
1개의 숫자, 문자, 구두점 등으로 컴퓨터가 읽거나 쓰는 기호이다. 모든 컴퓨터나 워드프로세서에서는 하나의 캐릭터는 1바이트로 기억되어 표현된다.

회로해석이 가능한 만능캐드
마이크로 캡(Ⅱ)

박일수/중앙대학교 전자공학과

서문

전기, 전자, 통신 및 그 관련학과에 다니는 학생이나 관련 직종의 실무자에게 있어서 전기회로와 전자회로의 해석은 가장 기본적이며 필수적인 일이다. 그러나 능동소자(TR, OP-AMP)가 포함된 회로의 이득이나 주파수 응답등을 구하는 것은 그리 쉬운 일이 아니다.

컴퓨터를 자주 사용하는 독자라면 한번쯤 프로그램을 작성해 보았거나 CAD용 유틸리티를 사용해 보았을 것이다.

기존의 CAD용 프로그램들은 주로 PCB기판을 디자인하기 위한 것으로 앞에서 언급한 회로해석은 거의 다루고 있지 않다.

그래서 필자는 위의 문제를 염두에 두어 마이크로 캡(MICRO CAP(Ⅱ))이라는 프로그램을 소개하고자 한다. 마이크로 캡은 세운상가등에서 쉽게 구할 수 있으며 사용법도 매우 간단하다. 또한 그래픽기능도 막강하여 CAD용으로도 손색이 없으며 특이한 점은 우리나라에 가장 많이 보급되어 있는 APPLE-IIe 혹은 APPLE-IIc에서도 프로그램 실행이 가능하다는 점이다. 좀 더 정확히 말하면 80칼럼 카드를 갖춘 APPLE-IIc에서 프로그램 실행이 가능하다는 것이다.

앞으로의 모든 설명은 16비트를 위주로 하겠지만 애플 사용자들도 사용하는데 어려운 점이 없을 것이다. 그러면 마이크로 캡 속으로 들어가 보도록 하겠다.

본문

(1) 셋업

●동작에 필요한 H/W

＊IBM-PC의 경우

–256KB 이상의 RAM

–2대의 FDD 또는 HDD

–적당한 그래픽카드

＊애플의 경우

–128KB 이상의 RAM

–바이덱스 카드

–2대의 FDD

＊이후부터 아무런 언급이 없으면 IBM-PC, 애플 공통의 내용이다.

●디스켓 구성

＊프로그램 디스켓 1장

＊데이터 디스켓 1장

●백업과 인스톨

애플의 경우는 백업본을 만들어 그대로 사용하면 되지만 IBM-PC의 경우, 다음과 같이 디스켓을 구성하면 효과적으로 프로그램을 사용할 수 있다.

```
A>FORMAT B:/S
```

S옵션을 이용하여 직접 부팅을 가능하게 한 후 디스켓 1번의 내용을 모두 복사한 다음 다음의 배치화일을 만들어 첨가한다.

```
A>COPY CON AUTOEXEC.BAT
GRAPHICS
SIMCGA1
DESIGNER
^Z
     1 file(s) copied
A>
```

컬러 그래픽카드를 구비하고 있는 독자라면 SIMCGA1을 생략하고, 허큘리스카드를 사용하고 있는 독자는 이외에 다른 MONO로 설정해 주는 프로그램을 이용해도 좋다.

프로그램 기동시 A드라이브에는 프로그램 디스켓이, B드라이브에는 데이터 디스켓이 들어 있어야 한다.

(2) 프로그램의 구성

그림 1과 같이 간단한 모듈로 구성되어 있다. 메인 프로그램인 DESIGNER에서 회로를 작성하고 저장하며 디스켓으로부터 데이터를 찾는 기능, 그리고 라이브러리를 관리하고 해석방식을 선택하는 중추적인 기능을 담당한다. 각 모듈들은 화일명칭 그대로이며 이후에 더 자세히 설명하게 된다.

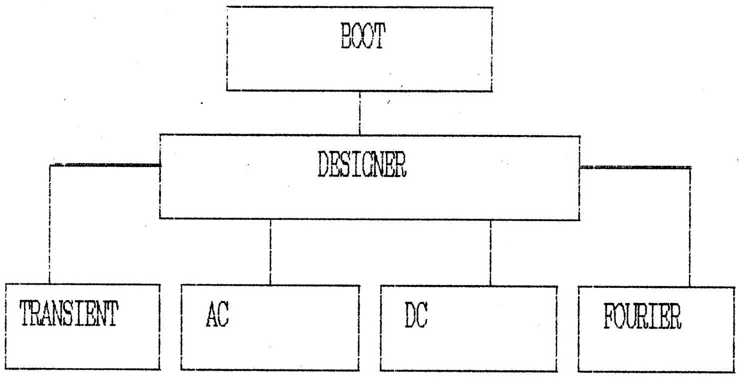
(3) 회로소자의 형태

마이크로 캡에는 모든 회로소자가 저장되어 있고 특별히 정의된 소자값이 라이브러리에 기록되어 있어서 회로상에서 간단히 라벨(Label)만 붙임으로써 특정하게 정의된 소자의 값을 이용할 수 있다. 참고로 저장된 소자와 갯수를 **표 1**에 나타내었고, **그림 2**에는 각 소자의 심볼과 약자를 나타내었다. 이 약자는 실제 회로를 그릴 때 사용하는 명령이 된다.

(4) 명령어 메뉴

프로그램이 부팅된 후 초기 화면은 **화면 1**과 같다. 어딘지 모르게 좀 허전한듯 하지만 곧 화면에 훌륭한 회로가 그려지며 작업을 수행하게 될 것이다.

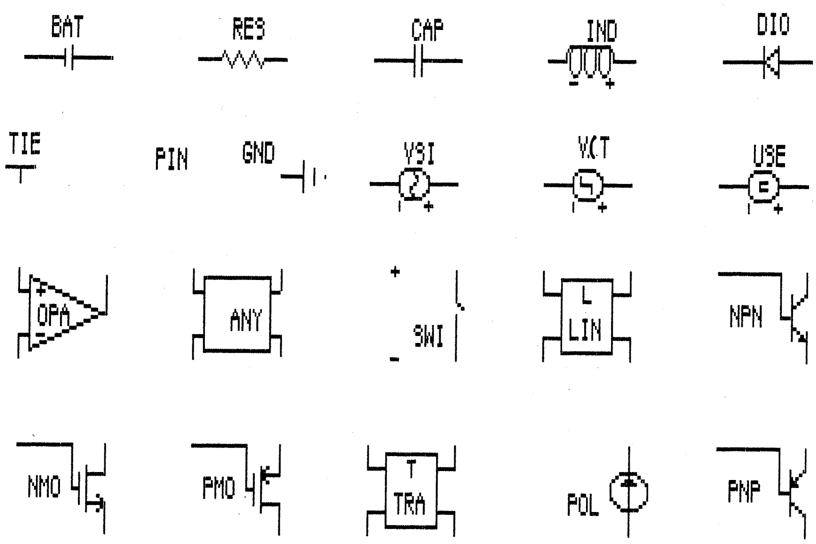
〔그림 1〕 마이크로 캡의 모듈구성



〔표 1〕 소자의 종류와 갯수

	소　　　　자	APPLE	IBM-PC
1	배터리(또는 전지)	10	10
2	저항 또는 레지스터	75	150
3	용량 또는 캐패시터	75	150
4	인덕터	25	50
5	사용자 소스	1	1
6	스위치	25	50
7	전송라인	5	5
8	다이오드	50	100
9	바이폴라 트랜지스터	25	50
10	변압기 또는 트랜스포머	10	25
11	MOS 디바이스	25	50
12	OP-앰프	15	30
13	사인파 전압소스	5	10
14	프로그램 가능한 전압소스	5	10
15	다항식 소스	15	25

〔그림 2〕 각 소자의 심볼과 약자



명령어를 실행시키려면 화면 첫줄에 보이는 명령어의 첫자를 키보드로 누르면 된다. 그럼 실제로 소문자 상태에서 'h'키를 입력해 보기 바란다. 예상대로 'Help'화면이 나타나야 하는데 아무런 반응이 없을 것이다. 이번엔 〈Caps-Lock〉키를 누른 후 대문자 모드에서 다시 'H'키를 입력해 보라. 명령어에 대한 요약된 내용이 화면에 나타났을 것이다. 이 실행에 의해 알 수 있듯이 마이크로 캡은 항상 〈Caps-Lock〉을 걸고 사용해야 하는데 이점은 꼭 기억해 두기 바란다.

'Help'화면에 나타난 명령어에 대해서 설명을 보충하면 다음과 같다.

- **I, M, J, K키** : 커서를 이동시킨다. IBM-PC에서는 화살표키를 사용해도 된다.
- **ENTER(E)키** : 회로의 소자를 그리기 위해 형태와 방향, 대칭, 값 등을 기록하는 기능을 갖는다.
- **ZAP(Z)키** : 잘못 그려진 소자를 수정하거나 아예 제거하는데 쓰인다. 'Z'키를 누르면 깜박이는 소자가 있는데 이때 커서이동키로 수정하고자하는 소자를 바꿀 수 있다.
- **SHORT(S)** : 이름 그대로 소자와 소자 사이를 연결시켜 준다.
- **BEGIN(B)** : 'ENTER'와 혼동하기 쉬우나 이 명령은 전혀 다른 기능을 수행한다. 항상 END(E)와 함께 쓰이며 'Begin Line'의 의미를 지닌다. 기능은 두개의 떨어진 노드(Node)를 연결시켜 주는 것인데 회로의 효율적 표현을 위해 꼭 필요한 명령이며 이후의 예제에서 그 기능을 잘 알게 될 것이다.
- **END(E)** :
  며 'BEGIN'명령으로 시작된 라인을 끝낸다. 'B-E'명령으로 그려진 라인은 'SHORT'명령때와는 달리 점선으로 그려지는데 이는 END명령이 내려진 노드 이외의 곳은 점선이 지났을지라도 SHORT되지 않은 것을 의미한다.

- DRAW(D) : 메모리 내에 있는 회로를 그려준다. 한번 그려진 회로는 'CLR' 명령에 의하지 않고는 지워지지 않는다. 프로그램의 실행중 회로가 없어졌더라도 'D'키를 누르면 다시 화면에 나타나게 된다. 그리 많이 쓰이는 키는 아니나 기억해 두었거나 유용하게 쓸 수 있다.
- NODES(N) : 그려진 회로의 노드에 번호를 할당해 줌으로써 해석에 편리하도록 한다.

회로를 그리고 해석할 때는 자동적으로 노드번호가 주어지지만 'B-E' 명령을 사용했을 때는 직접 'NODE' 명령을 사용해야 한다.

- CLR(C) : 메모리 내의 회로를 지운다. 중요한 회로를 그렸을 때는 이 명령을 사용하기 전에 디스켓에 저장하는 습관을 가져야 한다.
- FILE(F) : 디스켓으로부터 화일을 꺼내거나 저장하는데 사용한다.
- UTIL(U) : 각 유틸리티의 메뉴를 보여준다. 내장된 회로소자값을 프린터로 출력시켜 주며 데이터디스켓이 들어있는 드라이브의 전환도 해준다.
- LIB(L) : 내장된 모든 라이브러리를 보여준다. 이곳에서 독자들은 사용하기 원하는 소자를 찾을 수도 있고 원하는 값으로 편집할 수도 있다.
- PAGE(P) : 실제 쓰이는 방대한 회로의 해석에 없어서는 안될 명령어로써 페이지 전환을 해준다. 자세한 것은 예제를 통해 설명된다.
- QUIT(Q) : 마이크로 캡을 마치고 DOS 상태로 빠져 나온다.
- TITLE(T) : 사용자가 제목을 정의하여 붙일 수 있다.

(5) 예제

많은 유틸리티를 사용해 본 독자라면 매뉴얼에 있는 명령어 해설보다 예제를 실행시켜 보는 것이 이해하기에 더 쉽다는 것을 알 것이다. 필자자신도 예제가 이해를 하는 가장 확실한 방법임을 경험해 왔으므로 이후부터 4가지

〔화면 1〕 초기화면

```
Help Page Enter Zap Short Begin End Draw Nodes Clr File Analyze Util Lib Quit



          █
```

예제를 통하여 설명하고자 한다. 이 4가지 예제만 그대로 실행이 가능하다면 더 복잡한 회로를 실행하는데도 아무런 지장이 없을 것이라고 확신한다.

예제의 실행중에 이해가 되지 않는 부분이 있더라도 계속 설명이 되므로 두었다가 그대로 따라주기 바란다.

〈예제 1〉

가장 간단한 회로를 그리고 분석해 보기로 하자.

그림 3의 회로는 10V전압원과 2개의 100Ω짜리 저항으로 구성된 직렬회로이다. 그리는 순서는 다음과 같다.

```
E
GND ↵
D
```

그라운드(Ground)가 그려졌을 것이다.

```
E
V(T ↵
U
N
10V ↵
```

이번엔 10V 전압원이 윗쪽으로 그려졌을 것이다. 여기서 'V(T'를 'VCT'로 혼동하지 않기를 바란다. 알파벳의 'C'가 아니라 괄호의 '('를 나타내는 것이다.

```
E
RES ↵
R
N
100 ↵
E
RES ↵
D
N
100 ↵
```

두개의 100Ω짜리 저항이 직렬로 연결되어 그려졌을 것이다. 이제 저항을 그라운드에 연결하기만 하면 된다.

```
S
L
```

이제 회로가 완성되었다. 회로의 모양에 독자들은 아마 만족했을 것이다. 그러면 더 큰 기쁨을 맛보기 위해 해석모드로 가보자.

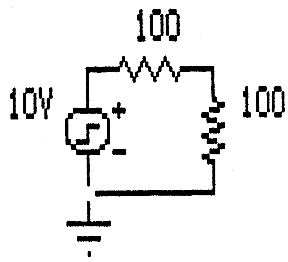
'A'를 누르면 Analysis option이 나온다. 3을 선택(DC)하면 화면 2가 나올 것이다. 여러가지 질문이 나열되어 있으나 일단 'Y'를 입력한다. 곧 화면 3이 나타날 것이다.

화면에서 보면 노드1에 10V전압이 걸릴 경우 노드2에는 5V전압이 걸리는 것을 알 수 있다. 따라서 이것은 전압분배법칙(Voltage divide rule)과 일치함을 알 수 있다.

이제 조금 복잡한 회로를 설계해 보도록 하자.

예제 2를 하기전에 'C'를 입력 해서 메모리를 클리어시킨다.

〔그림 3〕 예제 1의 회로그림



〈예제 2〉

이번에는 너무 기초적인 것을 벗어나서 전기회로의 핵심인 RLC회로를 마이크로 캡을 이용하여 해석해 보겠다.

해석할 회로는 그림 4에 나타나 있다. 그럼 시작해 보자. 자신이 있는 독자라면 필자의 방법과 다르게 시도해 보기 바란다.

```
E
V(T ↵
U
N
PULSE ↵
```

펄스 소스(Pulse Source)가 그려졌을 것이다. 'Parameter ?'라는 질문에 꼭 수치를 입력해야만 하는 것은 아니다. PULSE는 라이브러리속에 내장되어 있다.

```
E
IND ↵
R
N
1UH ↵
E
CAP ↵
D
N
1NF ↵
```

커서를 한칸 위로 올린다.

```
S
R
```

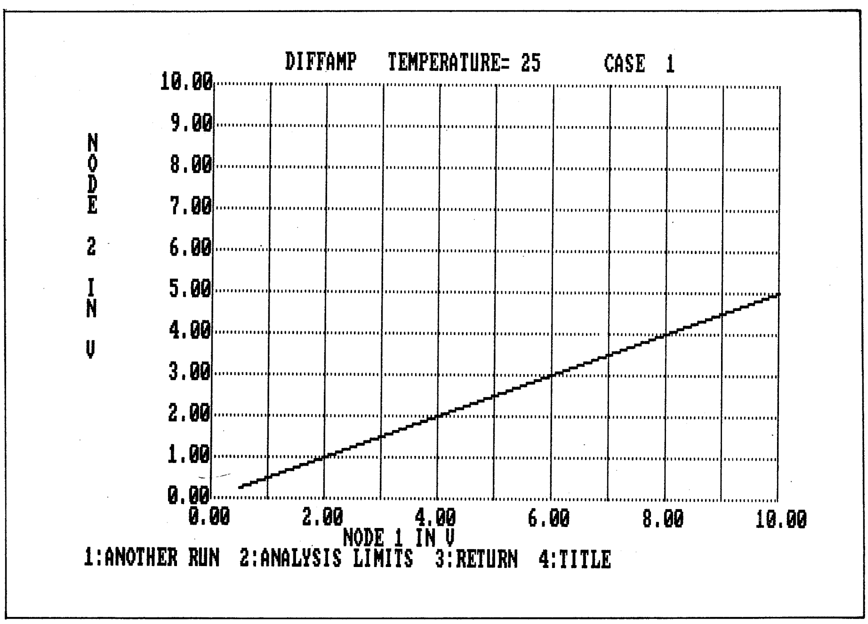
〔화면 2〕 해석옵션 화면

```
                    DC ANALYSIS PRLOGUE

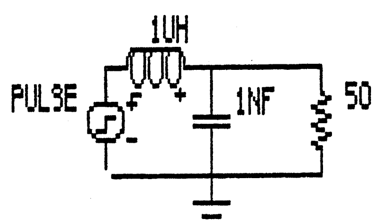
INPUT VOLTAGE RANGE(HIGH/LOW)          10
OUTPUT VOLTAGE RANGE(HIGH/LOW)         10
MAXIMUM INPUT VOLTAGE STEP             .5
INPUT NODE #                           1
OUTPUT NODE #                          2
NO. OF ITERATIONS                      50
PRINTED TABLE OF VALUES                N
EDIT INITIAL NODE VOLTAGES             N
TEMPERATURE (HIGH/LOW/STEP)            25
WORST CASE (Y/N)                       N
ERROR CRITERIA                         .001
RELAXACTION FACTOR                     2

                    ARE THESE CORRECT (Y/N) ?
```

〔화면 3〕 회로해석 그래프



〔그림 4〕 RLC회로 그림



```
E
RES ↵
D
N
50 ↵
S
L
E
GND ↵
D
S
L
```

회로가 그림 4와 같이 그려졌을 것이다. 실수에 의해 수치가 잘못 입력되었으면 'ZAP'기능을 이용하여 수정하도록 한다. 이 회로는 예제 1 의 회로와는 달리 과도현상과 교류의 해석이 매우 중요하다.

우선 과도상태(Transient)를 해석해 보기로 하자. 'A'키를 누른 후 옵션이 나오면 '1'을 선택한다. 잠시후 **화면 4**가 나오는데 일단 'Y'라고 입력한다. 그러면 **화면 5**의 펄스에 대한 과도응답이 그래프로 나타난다.

다음은 교류해석을 해석해 보도록 하겠다. 방법은 이미 독자들이 충분히 파악했을 것이다. **화면 6**과 **화면 7**을 확인해 보라. 화면 7은 회로의 안정도를 판별하는데 편리한 BODE PLOT이다. 이제까지 두 예제를 통하여 간단한 사용방법과 해석법을 익혔다. 더 진도를 나가기 전에 잠깐 몇가지의 주의사항과 정리할 점을 요약해 보았다.

- 키보드는 〈Caps-Lock〉을 걸고 사용해야 한다.
- 한가지 해석을 하고 메인 매뉴로 돌아온 후에도 회로는 메모리속에 있으므로 'D'를 누르면 다시 나타난다.
- 새로운 회로를 그릴때는 'C'를 눌러 메모리를 지운 뒤에 한다.
- 소자의 명칭을 입력할 때 그림 2와 같이 3자만 입력한다. 특히 'V(T')의 표기에 조심한다.

〔화면 4〕 과도상태 해석을 위한 화면

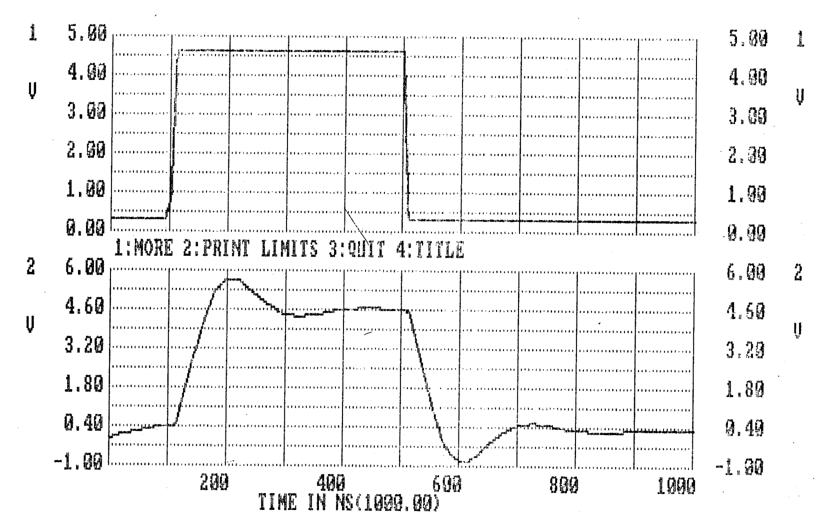
```
 NETWORK DIFFAMP...ITEMS= 8
RESISTORS.................... 2
CAPACITORS................... 1
INDUCTORS.................... 1
V(T)......................... 1
NETWORK SIZE IN MODES= 2


                              ANALYSIS LIMITS

MAXIMUM SIM TIME (SEC)                  1E-6
MAXIMUM DELTA TIME (SEC)                1E-8
MINIMUM ACCURACY (%)                    5
UPPER TRACE A                           1
UPPER TRACE B                           1
UPPER TRACE RANGE(HIGH/LOW)             5
LOWER TRACE A                           2
LOWER TRACE B                           2
LOWER TRACE RANGE(HIGH/LOW)             6/-1
ZERO INITIAL CONDITIONS (Y/N)           Y
EDIT/REVIEW INITIAL CONDITIONS (Y/N)    N
DUMP NODE WAVEFORM TO USER FILE (Y/N)   N
CRT PLOT(C) OR TABLE(T)                 C
WORST CASE (Y/N)                        N
TEMPERATURE (LOW/HIGH/STEP)             25
CALCULATE D.C. OPERATING POINT(Y/N)     N

ARE THESE CORRECT (Y/N) ?
```

〔화면 5〕 펄스에 대한 과도응답 그래프

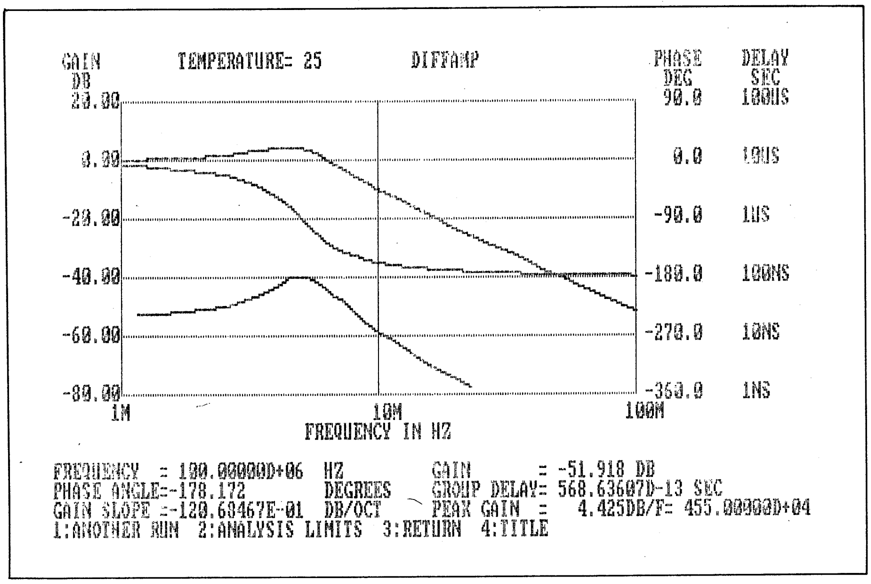


〔화면 6〕 옵션선택 화면

```
ANALYSIS LIMITS

LOWEST  FREQUENCY                 1E6
HIGHEST FREQUENCY                 1E8
LOWEST  GAIN(DB)                  -80
HIGHEST GAIN(DB)                  20
LOWEST  PHASE SHIFT               -360
HIGHEST PHASE SHIFT               90
LOWEST  GROUP DELAY               1E-9
HIGHEST GROUP DELAY               1E-4
INPUT  NODE NUMBER                1
OUTPUT NODE NUMBER                2
ACCURACY(%)                       5
PRINT TABLE OF VALUES(Y/N)        N
AUTO OR FIXED FREQUENCY STEP      A
DEFAULT PLOTTING PARAMETERS(Y/N)  Y
TEMPERATURE (LOW/HIGH/STEP)       25
WORST CASE (Y/N)                  N
ARE THESE CORRECT (Y/N) ?
```

〔화면 7〕 회로의 안정도를 판별하는데 편리한 BODE PLOT



• 소자값을 입력하는 방법은 2가지가 있다. 하나는 직접 K(10^3), U(10^{-6}), P(10^{-9}) 등의 단위로 입력하거나 U나 P대신 'E-6', 'E-9'를 사용한다. 다른 하나는 저항의 경우는 R1, R2, …로 하고 트랜지스터의 경우는 Q1, Q2,…로 라벨을 붙이고 라이브러리에 그 값을 정의하여 저장한 후 사용하는 방법이다.

• 디렉션과 리플렉션에 관한 것인데 디렉션은 방향의 첫째 문자를 누르기만 하면 된다. 리플렉션은 X, Y, None의 선택이 있는데 None를 선택('N'을 누름)하면 그림 2의 그림이 그대로 그려지고 X나 Y를 선택하면 X축과 Y축에 대칭으로 그려진다.

이제 능동소자가 들어 있는 회로로 들어가 보도록 하자.

〈예제 3〉

능동소자가 들어 있는 회로 해석

먼저 그림 5의 회로를 그려보기 바란다. 필자는 다음과 같은 방법으로 회로를 그렸다.

```
E              10K ↵
GND ↵          E
D              NPN ↵
E              R
V(T ↵          N
U              Q2 ↵
N              E
PULSE ↵        RES ↵
E              U
RES ↵          N
R              500 ↵
N
```

커서를 에미터(Emitter)까지 내린다.

```
S
L
S
L
```

커서를 다시 에미터로 옮긴다.

```
S              10V ↵
R              S
E              U
BAT ↵          S
U              L
N
```

독자들이 직접 이 회로를 해석해 보기 바란다. 여기서 한가지 매우 중요한

기능을 알아야 하는데 트랜지스터 회로는 온도 등의 주변환경에 민감하므로 보통은 트랜지스터만의 회로는 사용되지 않고 피드백 회로로 구성한다. 즉 저항이나 캐패시터를 입력과 출력 사이에 연결한다. 그래서 그림 5의 회로를 보면 Q2의 콜렉터단자와 그라운드 사이에 100PF의 캐패시터를 연결하려고 하는 것이다. 그럼 어떻게 하면 되는가? 그림에서 보듯이 10V 배터리와 트랜지스터 사이가 너무 좁으므로 캐패시터를 외부에 설치해야 한다. 필자가 제시하는 방법대로 한번 해보기 바란다.

멀리 떨어진 두 노드 사이를 연결하거나 그 사이에 소자를 삽입시키는 방법은 2가지가 있다.

〈방법 1〉

그림 2의 'TIE'를 사용한다. 즉 커서를 콜렉터 연결점에 위치시킨 후 다음과 같이 입력한다.

```
E
TIE ↵
R
C ↵
```

커서를 10V 배터리 그라운드 쪽에 위치시킨다.

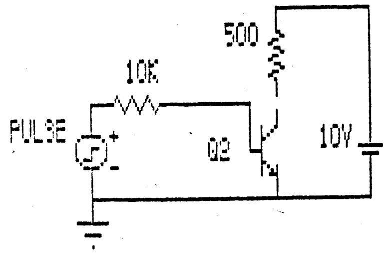
S	100PF ↵
R	
	E
E	TIE ↵
CAP ↵	U
U	C ↵
N	

이제 그림 6과 같이 되었을 것이다. 두점은 모두 'C'라고 되었으며 아무 선으로도 연결되지는 않았다. 그렇지만 실제 해석시 이 두 지점은 연결된 것으로 해석된다.

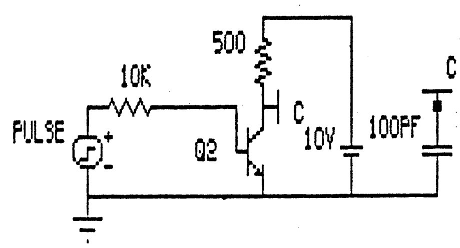
〈방법 2〉

이 방법에 들어가지 전에 자기가 그

〔그림 5〕 예제 3의 회로그림



〔그림 6〕 두 노드 사이의 연결방법(1)



린 회로를 저장하는 방법에 대해 잠깐 살펴보자. 방금 만든 회로를 'TEST'란 이름으로 저장해 보기로 하자.

```
F
2
TEST ↵
```

방법 2를 하기 위해 먼저 회로를 캐패시터가 없도록 그림 5와 같이 원상으로 한다. 'ZAP'기능을 비로소 사용할 때가 왔다. 'ZAP'기능을 사용하지 못하겠다고 생각되면 회로를 지우고 다시 그려도 좋다. 좀 힘들겠지만…

먼저 그림 5의 10V 배터리의 그라운드쪽에 커서를 위치시키고 다음과 같이 입력한다.

```
S
R

E
CAP ↵
U
N
100PF ↵
B
```

커서를 콜랙터까지 이동한다.

```
E
```

여기서 E는 End Line을 의미한다. 즉, 앞에서 설명했던 'B-E'명령으로 **방법 1**과 같은 효과를 얻은 것이다. 두번째 방법이 시각적으로 편리한 것 같지만 복잡한 회로에서는 상당히 혼동을 주기 쉽다. 그러므로 적절히 방법 1과 방법 2를 조합해서 사용하는 것이 바람직하다. 방법 2의 결과는 **그림 7**에 나와 있으니 확인 해보기 바란다. 'N'을 누르면 노드번호가 붙여진다.

〈예제 4〉

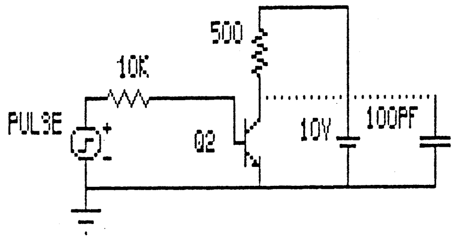
앞에서 몇가지 예제를 실행시켜본 결과가 만족스러웠는지 모르겠다. 혹시 만족스럽지 않았더라도 **예제 4**를 실행해 보면 그 마음이 달라질 것으로 안다. 먼저 그림 8을 그려보기 바란다. 그림 8과 똑같이 그려낸 독자가 있다면 그 독자는 나머지 부분을 읽지 않아도 좋을 것이다. 하지만 처음에 성공할 수 있을런지 의심스럽다. 아마도 실패한 주 원인은 회로 네구석의 A, B, C, D를 쓰는 문제일 것이다.

그림 2를 보기로 하자. 이미 배웠거나 실행시켰던 것인데 한가지 생소한 것이 있다. 바로 'PIN'이다. 이것이 위의 문제를 해결하는 열쇠가 된다.

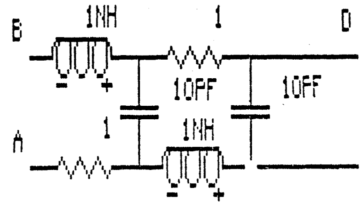
다음과 같은 방법으로 그림 8의 회로를 그려보자. (왼쪽 아래부터 그린다.)

E	IND ↵
PIN ↵	R
A ↵	N
	1NH ↵
E	
RES ↵	S
R	R
N	
	E
1 ↵	
	PIN ↵
E	C ↵

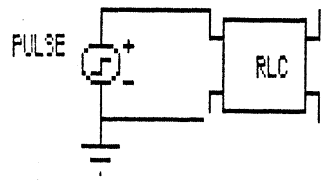
〔그림 7〕 두 노드 사이의 연결방법(2)



〔그림 8〕 예제 4 의 회로그림



〔그림 9〕 BOX표현 예



커서를 왼쪽위로 옮긴다.

E	RES ↵
PIN ↵	R
B ↵	N
	1 ↵
E	
IND ↵	S
R	R
N	
1NH ↵	E
	PIN ↵
E	D ↵

이제 남은 일은 두개의 캐패시터를 그리는 일이다. 커서를 IND와 RES사이로 옮긴 후 다음과 같이 입력한다.

```
E
CAP ↵
D
N
10PF ↵
```

커서를 오른쪽으로 한칸 옮긴다.

```
E
CAP ↵
U
N
10PF ↵
```

회로가 완성되었다. 여기서 한가지 궁금한 점이 또하나 있다. 왜 복잡하게 A, B, C, D 라벨을 붙였을까. 그 답은 곧 나올 것이다. 우선 다음과 같이 조작하여 회로를 디스켓에 저장한다.

```
F
2
RLC ↵
```

다음엔 'C'를 눌러 메모리를 지운 다음 다시 불러낸다.

E	S
GND ↵	R
D	
	E
E	RLC ↵
V(T ↵	R
U	N
PULSE ↵	

여기까지 실행했으면 위에서 왜 라벨을 붙였는지 알 수 있을 것이다. 즉 복잡한 회로가 간단히 BOX로 표현된 것이다. 계속하여 그림 9 의 회로를 완성시켜 보자.

커서를 왼쪽 아래 단자로 옮긴 후 다음을 입력한다.

```
S
L
```

그림 9 와 동일한 회로가 완성되었다. 전기, 전자회로를 이미 잘 아는 사람이라면 '4단자망 회로'라는 것을 알고 있을 것이다. 여기서 만일 OP-AMP와 같은 3단자로 된 BOX를 만들고자 한다면 두개의 단자를 Short시키면 될 것이다. 이와같이 사용자가 임의로 회로를 정의하여 사용할 수 있으므로 자주 사용하는 회로(증폭, 발진)를 미리 정의해 놓고 매크로로써 사용하면 매우 효율적이다(이와 같은 사용자 정의 회로를 User Defined Network이라 한다.).

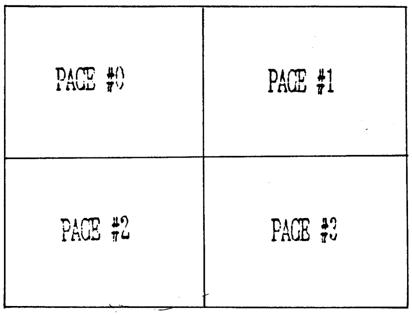
이제부터는 마이크로 캡의 다른 기능에 대해 알아 보자. 만약에 회로가 방대하여 모니터 화면을 다 채우고도 남아 더 이상 그릴 수 없다면 어떻게 할 것인가? 앞에서처럼 BOX에 넣을 수도 없는 일이다. 이점에 대해서는 마이크로 캡은 좋은 환경을 제공해 주고 있다. 즉 4개의 페이지를 갖추고 있는데 그것은 그림 10과 같다. 즉 프로그램 부팅 후 초기 화면은 페이지 0이고 우측은 페이지 1, 아래쪽은 페이지 2가 된다. 커서를 맨 오른쪽으로 옮기면서 실제로 확인 해보기 바란다.

```
E
OPA ↵
R
N
709 ↵
```

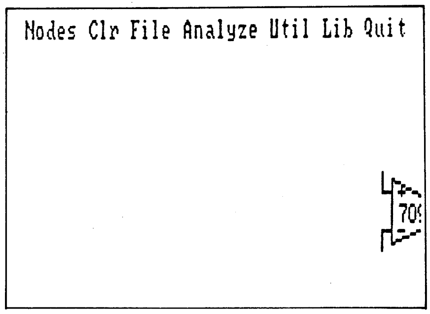
화면 8과 같이 될 것이다. 그렇다고 회로가 잘린것은 아니므로 안심하자. 다시 다음을 입력해 보기 바란다.

```
P
1 ↵
```

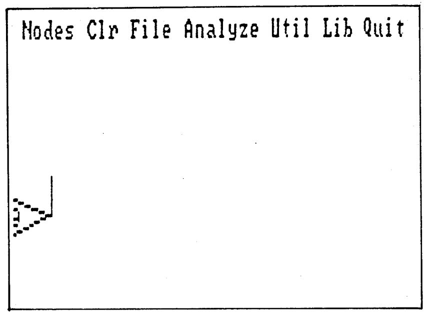
화면 9와 같이 되었을 것이다. 즉 회로의 연속성이 확인되었다. 화면 4개를



〔그림 10〕 4 페이지로 분리된 마이크로 캡 화면



〔화면 8〕 페이지 0



〔화면 9〕 페이지 1

모두 사용한 후 회로를 복사하려면 페이지0, 페이지1, 페이지2, 페이지3를 각각 하드카피하여 잇는 수밖에 없다.

(6) 보다 효과적인 활용을 위하여

마이크로 캡은 이미 살펴본 바와 같이 메뉴방식으로 되어 있어서 라이브러리, 유틸리티, 회로해석 프로그램 등을 이용하는데 큰 어려움은 없으리라 본다. 그래서 필자가 힘겹게 느꼈던 부분만을 골라 간략하게 설명하겠다.

●라이브러리의 활용

앞 부분의 예제에서도 언급했지만 파라미터는 수치뿐만 아니라 라벨로도 가능하다고 했다. 그 이유는 저항, 캐패시터, 인덕터 등의 수동소자의 경우는 곧바로 수치기입이 용이하나 능동소자의 경우는 그렇지가 않다. 예를 들어 OP-AMP의 경우 다음의 8가지 사항이 그 특징을 결정해 버린다.

*입력저항
*출력저항
*개루프 이득
*입력 오프셋 전압
*최대 출력 전압
*First Pole(Hz)
*Second Pole(Hz)
*Slow Rate(Volt / Sec)

이러한 사항을 매번 재입력한다는 것은 매우 귀찮은 일이다. 그러므로 라이브러리에 저장해 놓으면 편리하게 사용할 수 있다는 것이다. 표 1에서와 같이 마이크로 캡은 여러 종류의 라이브러리를 내장하고 있으므로 거의 새로운 입력없이 기존 입력된 내용을 종종 찾아 쓰면 되는 것이다.

또 한가지 짚고 넘어갈 것이 있다. 'L'을 눌러 라이브러리를 선택하고 'P'를 눌러 저항을 선택하여 보라. 아마 수치는 다르더라도 다음과 같은 형식으로 되어 있을 것이다.

```
70K/10/1000
200/10/1000
```

이것은 다음과 같은 것을 의미하고 있으며 반드시 '/'로 각 요소들을 구분하고 있어야 한다.

```
VALUE/TOLERANCE IN PERCENT/
TEMPERATURE COEFFICIENT IN PPM
```

이외에 소자값의 입력은 특별한 것이 없고 화면에 나타난 특성에 맞게 값을 수정하거나 새로운 라이브러리를 만들면 된다.

●유틸리티의 활용

'U'를 입력하면 아주 간단한 메뉴가 나온다. 하지만 이 기능은 실제 회로를 설계하는 사람에게는 상당히 유용하게 이용될 수 있다.

첫번째 기능은 현재 분석한 회로의 각 소자의 값과 특성을 프린터로 출력해 준다. 부품이 옳게 들어가 있는지, 올바른 대치품을 사용했는지를 쉽게 확인해 주기도 한다.

두번째 기능은 드라이브 설정을 바꾸어 준다. 일반적으로 데이터 디스켓은 B에 있도록 세트되어 있으나 이 기능을 이용하면 C나 D로도 바꿀 수 있다.

●해석 옵션

앞의 예제에서는 AC, DC 및 트랜견트(Transient)를 해석할 때 나오는 옵션을 그대로

```
ARE THESE CORRECT (Y/N) ?
```

라는 질문에 'Y'로만 입력하여 실행시켰다. 이유는 비교적 간단한 회로에서는 자동적으로 가장 적당한 옵션으로 변환되기 때문이다. 그러나 복잡한 회로에서는 이 옵션들을 어떻게 선택하느냐에 따라 보다 정확한 응답을 얻을 수 있다. AC, DC나 트랜젼트 옵션중 트랜젼트 경우만 알면 나머지 AC나 DC는 쉽게 응용할 수 있으므로 트랜젼트에 대해서 살펴보기로 하자. 역시 설명을 위해서 예가 필요하므로 이제까지 익힌 것을 바탕으로 다음을 입력해 보기 바란다.

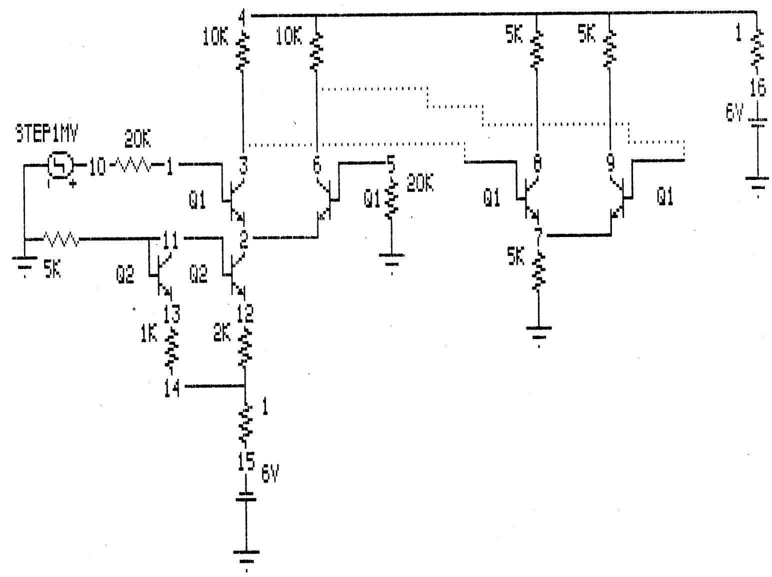
```
F
1
DIFFAMP ↵
5 ↵
```

그림 11이 나타났을 것이다. 이제는 트랜젼트 해석을 선택한다.

```
A
1
```

각 노드마다 번호가 붙여지고 잠시 후 화면 10의 옵션들이 나타난다. 이것

〔그림 11〕 AC, DC 및 트랜젼트 해석을 위한 회로예



〔화면 10〕 트랜젼트 해석을 위한 옵션

```
                         ANALYSIS LIMITS

MAXIMUM SIM TIME (SEC)                      1E-5
MAXIMUM DELTA TIME (SEC)                    1E-7
MINIMUM ACCURACY (%)                        5
UPPER TRACE A                               10/0/V
UPPER TRACE B                               3/6/V
UPPER TRACE RANGE(HIGH/LOW)                 .001/0/0/-.04
LOWER TRACE A                               8/9/V
LOWER TRACE B                               14/15/0/P
LOWER TRACE RANGE(HIGH/LOW)                 2/0/.01/-.01
ZERO INITIAL CONDITIONS (Y/N)               Y
EDIT/REVIEW INITIAL CONDITIONS (Y/N)        N
DUMP NODE WAVEFORM TO USER FILE (Y/N)       N
CRT PLOT(C) OR TABLE(T)                     C
WORST CASE (Y/N)                            N
TEMPERATURE (LOW/HIGH/STEP)                 25
CALCULATE D.C. OPERATING POINT(Y/N)         N

ARE THESE CORRECT (Y/N) ?
```

은 앞에서 본 화면 4의 옵션보다 복잡한 형식이 첨가된 것이다. ('/', '/V' 등)이 옵션들에 대해서 설명하겠다.

첫째줄의 옵션은 최대 시뮬레이션 시간을 표시한다. 둘째줄의 것은 시뮬레이션 중 시간의 증가량을 표시한다. 세째줄은 DELTA T의 변화를 자동으로 수정해주는 정도를 표시하며 일반적으로 1~5%로 한다. 위의 3가지 옵션들은 가능하면 바꾸지 않고 쓰는 것이 좋다. 중요한 것은 네번째에서 아홉번째 줄에 있는 것이다. 여기서 일단 'Y'를 눌러 해석을 시킨 다음 화면 11를 얻도록 하고 화면 10과 비교하면서 다음의 설명을 보면 이해하는데 도움이 될 것이다. 화면 11을 보면 윗 부분에 2개, 아랫쪽에 2개의 트레이스(Trace) 한 결과가 나타나 있다. 즉 위쪽의 2개의 결과는 UPPER TRACE A와 B이고, 아랫쪽의 것은 LOWER TRACE A와 B이다. 예로 'UPPER TRACE A 10/0/V'는 노드 0과 10 사이의 전압을 트레이스했다는 것을 나타낸다. 'LOWER TRACE B 14/15/O/P'는 노드 14와 15사이의 전력을 트레이스 했다는 것이고 '0'는 무의미한 것이다. 이를 정리해 보면 다음과 같다.

```
NODE #/NODE#/V or P or E or I
```

V : 전압
P : 전력
E : 에너지
I : 전류

다음의 경우는

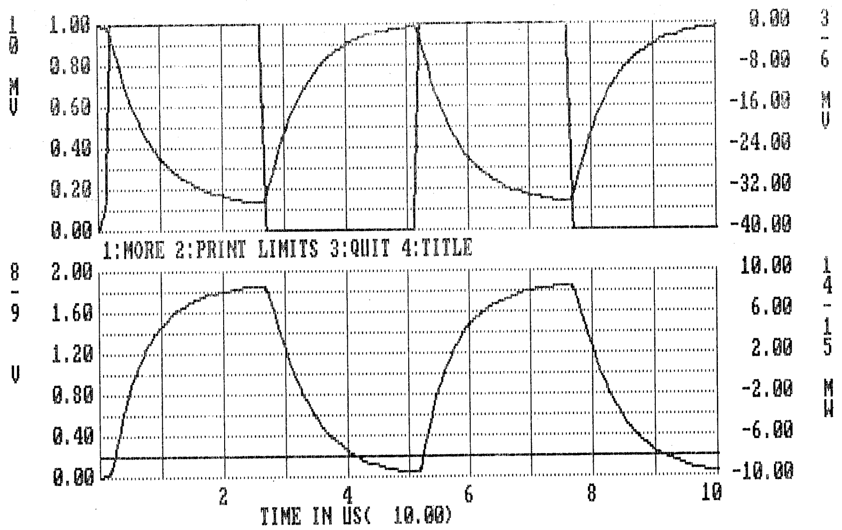
```
UPPER TRACE RANGE(HIGH/LOW)─┐
                  └→.001/0/0/-0.4
```

트레이스하는 RANGE를 뜻한다. 앞의 두개는 A의 범위, 뒤의 두개는 B의 범위이다. 즉 트레이스 A는 0V~0.001V, 트레이스 B는 −0.4V~0V의 범위에서 측정한다는 것이다. 이를 정리하면 다음과 같다.

```
HIGH A/LOW A/HIGH B/LOW/B
```

이제 열번째 줄의 옵션에 대해서 알아보자. 이것은 회로의 모든 소자가 초기 조건이 0이면 'Y'로 응답하고 그렇지 않으면 열한번째 옵션으로 수정하면 된다. 열네번째 'OPTION WORST CASE(Y/N)?'에 대한 설명으로 옵션에 대한 소개를 마치겠다. 이것은 소자의 톨러런스(Tolerance)에 대한 내

〔화면 11〕UPPER TRACE와 LOWER TRACE



용으로 다음의 범위내에서 사용자가 지정해 준 횟수만큼 실행되면서 톨러런스의 변화에 따른 응답의 변화를 보여준다.

```
VALUE low = NOMINAL*(1-tolerance/100)
VALUE high = NOMINAL*(1+tolerance/100)
```

이상으로 옵션에 대한 내용을 알아 보았다. 끝으로 옵션의 내용은 노드의 변화외에는 될 수 있는한 고치지 말고 디폴트의 값을 그대로 사용하였으면 하는 것이 필자의 생각이다. 그리고 범위(Range)도 가능한 작게 조절하는게 좋다.

결문

이제까지 부족한 설명이나마 마이크로 캡에 대해서 소개하였다. 그리고 설명에서 푸리에 해석은 생략하였는데 급수해석은 수학적 설명이 많이 필요하고 지나치게 전문적인 방향으로 치닫게 되는 경향이 있기 때문이다. 관심이 있는 독자는 필자에게 연락을 하면 성실히 답해 줄것을 약속한다.

'88년 12월호 바이트(BYTE)지를 보니 MICRO-CAP(Ⅲ)에 대한 기사가 있었는데 MICRO-CAP(Ⅱ)만 제대로 다룰줄 알면 신버젼인 CAP(Ⅲ)를 구입하게 되어도 무난하리라 본다. PC

애독자 선물대잔치

2월호 당첨자 명단

●디스켓 박스(10장들이)●

윤태원 서울시 강남구 역삼2동 개나리 APT 19-402
이도원 서울시 도봉구 쌍문1동 306-30 29/9
이화순 전북 완주군 구이면 계곡리 377번지
정석구 부산시 사하구 당리동 339-6 16/3 미아맨션 109호
최명숙 경기도 안양시 호계동 주공 APT 221동 208호

●디스켓 보관함●

김상훈 서울 강동구 명일동 삼익 그린 APT 603-612
신동주 경남 울산시 남구 신정 3동 541-7
윤선경 서울 서대문구 북가좌2동 서도시범주택 나동201호
이재훈 서울 강동구 길동 길동APT 13동 401호
한유경 대구시 서대구 우체국 사서함 20호 무스교역상사

●컴퓨터 용어사전●

김진상 광주시 서구 광천동 57-9 20/3
김학준 서울 강남구 역삼 2동 개나리 APT 14동 205호
남진배 경남 울산시 남구 야음2동 574-4 장수 APT 102호
이상기 대구시 달서구 감삼동 175-5
황승진 서울 서대문구 남가좌2동 80-56

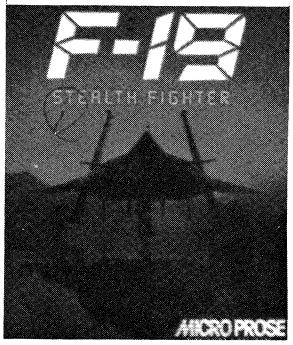
●디스켓 자켓(4장들이)●

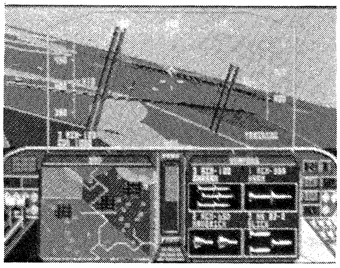
강영덕 제주도 북군 애월읍 장전리 1087번지
남승현 서울 양천구 목4동 784-9
윤성문 부산시 금정구 부곡1동 847-7 9/5
이광재 서울 마포구 염리동 65-8 8/6
정세균 인천시 중구 항동7가 1-2 인천시경 기동 1중대
정재헌 서울 강서구 화곡4동 470-15
정한신 서울시 도봉구 미아4동 미아 APT 2동 105호
조영일 경기도 수원시 성균관 대학교 물리학과 2년
한경태 대전시 중구 호동 240-3번지
현 솔 강원도 원주시 봉산1동 1088-7 (원용인씨댁)

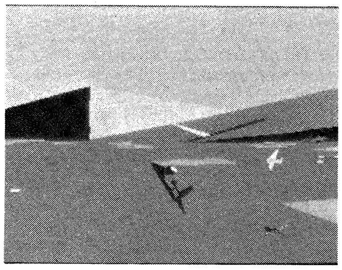
●IBM-PC 입문●

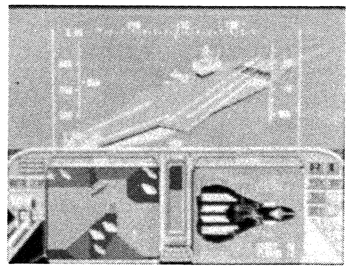
고동명 부산시 서구 아미동 2가 95 7/5
서광석 충남 천원군 성환읍 성월리 144 성환주택 가-102
신원식 서울 성동구 용납동 121번지
조정욱 부산시 금정구 장전3동 643-355 9/2
조형래 김천시 지좌동 753번지

※ 서울 거주자는 3월31일까지 신분증을 지참하시고 직접 본사로 나오시면 해당 상품을 드리며, 지방에 계신 분은 우송해 드리겠읍니다. 기간은 3월31일까지이며, 이후는 무효로 처리합니다.

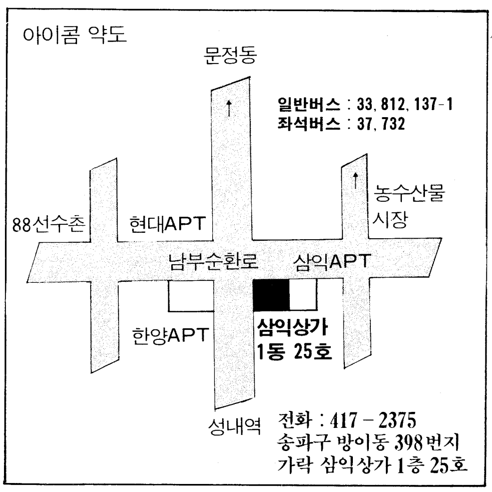








10TH FRAME PROSOWLING
221B BAKER STREET
3D HELICOPTER
ACE
ACE OF ACE
AIR TRAFFIC CONTROLLER
ALTERNATE REALITY
ANCIENT ART OF WAR AT SEA
ARTIC FOX
ATLAS
BAD STREET BRAWER
BALANCE OF POWER
BARD'S TALE I
BATTLE OF MIDWAY
BATTLE ZONE
BEYOND CASTLE WOLFENCTEIN
BIONIC COMMANDO
BLACK CAU;DRON
BLACK JACK
BLACK JACK ACAEMY
BLOCK BUSTER
BOP,N WRESTLE
BOORROWED TIME
BREACH
BUBBLE GHOST
BUCK ROGERS
BY FIRE OF SWORD
CALIFONIA GAME
CANFIELD CARD GAME
CHAMPIONSHIP BASEBALL
CHAMPIONSHIP BOXING
CHAMPIONSHIP GOLF
CHUCK YEAGER'S FLIGHT TRAINNER
COMMANDO
DALLAS QUEST
DAM BUSTER
DARK CASTLE
DEATH SWORD
DEFENDER OF THE CROWN
DESTROYER
DRAGON WORLO
EAGLE
EARL WEAVER BASE BALL
EMPIRE
F-15 STRIKE EAGLE
F-19 STEALTH FIGHTER
FALCON F-16
FAMILI FOED
FIRE POWER
FIRST EXPEDOTION
FLIGHT SIMULATOR
FLIGHT SIMUATOR V3.0
FORD SIMULATOR
FRIENDLY WARE GAME
FULL COUNT BASEBALL
GAMESTAR BASEBALL
GATO
GAUNTLET
GHOSTS'N GOBLINS
GUNSHIP
HACKER
HARD BALL
HARRIER
HEROES OF LANCE
HOSTAGE
HUNT FOR RED OCTOBER
IKARI WARRIOR
IMAGE TENNIS
INDOOR SPORTS
INFILTLARTOR
INFOCOMIC
INTO THE EAGLE'S NEST
JACK NIKLAUS GOLF
JEOPARDY
JET
JEWEL'S OF DARKESS
KARNOV
KING'S QUEST
KING'S QUEST
KING'S QUEST
KING'S QUEST
KNIGHT ORC
LAST NINJA
LEISURE SUIT LARRY II
LOCK ON
LODE RUNNER
MAD SPY VS SPY III
MAN HUNTER
MARBLE MADNESS
MASTER NINJA
MIGHT & MAGIC
MINI PUTT
MIXED UP MOTHER GOOSE
MOEBIUS
NEW SOKO BAN
NFL CHALLENGE
NINJA
OFF SHORE WARRIOR
ONE ON ONE
PATTON VS ROMMEL
PETE ROSE PENANT EFVER
PHANTASIE
PHANTASIE III
PHYCHO
PINBALL WIZARD
PINNBALL CONSTRUCTION KIT
PILICE QUEST
POLICE QUEST II
POOL OF RADIANCE
PRLEAGE SPORTS
PT-109
QUSTRON II
RAD WARRIOR
RAMPAGE
RINGS OF ZILFIN
ROADWAR 2000
SDI
SEARCH AND DISTROY
SHOGUN
SILENT SERVICE
SKATE OR DIE
SKY & AIR NAVIG
SKY RUNNER
SPACE QUEST I
SPACE QUEST II
SPIDER MAN
STAR COMMAND
STAR FLEET I
STAR FLIGHT
STAR GLIDER
SUB BATTLE
SUMMER GAME II
SUPEBYKE
SUPERSTAR ICEHOCKEY
SWORDS OF CLASS
TEED OFF CLASS
TEED OFF (GOLF GAME)
TEST DRIVE
TETRIS
THE GANES SUMMER GAME EDITION
THE LAND OF THE LOUNGE LIZARDS
THEXDER
THREE STOOGES REEL
TOP GUN
TREASURE ISLAND
TROHAN
ULTIMA I
ULTIMA II
ULTIMA III
ULTIMA
ULTIMA
WHERE IN THE EUROPE IS CARMEN
WHERE IN THE USA IS CARMEN
WHERE IN THE WOOLRD IS CARMEN
WINTER GAMES
WIZARD WARS
WIZARDRY
WORLD GAME
WORLO KARATE CHAMPIONSHIP

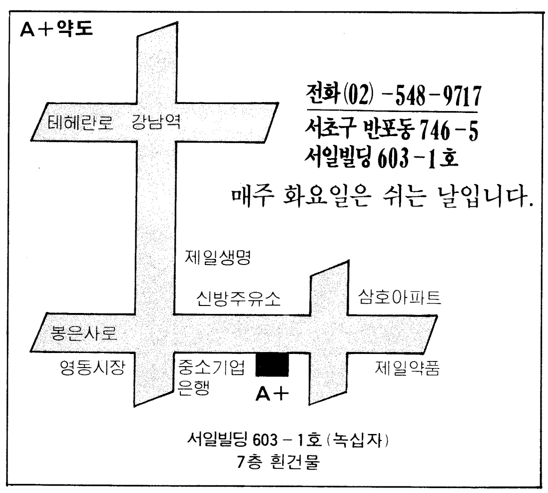


회원모집 안내

년가입비 : 10,000
월회 비 : 10,000

지방 USER안내

1. 지방회원은 송료만 추가하시면 됩니다.
2. 필요한 SOFT WARE는 소액환으로 보내주시면 프로그램을 보내드립니다.



타이머

모든 IBM-PC는 인텔 8253이나 8254 타이머칩을 사용하여 시스템 클럭칩에서 출력되는 펄스를 기록한다. 보통 시스템클럭의 사이클은 단일 펄스로 변환되며, 이렇게 단일 펄스로 변환된 펄스의 체인은 타이밍을 표현하기 위해 카운트하거나 컴퓨터의 스피커가 특정 주파수의 사운드를 낼 수 있도록 하는 역할을 한다. 8253칩은 각기 독립적인 3개의 채널을 갖고 있으며 이 채널은 각각 개별적으로 제어가 가능하다.

8253칩은 CPU와 독립적으로 동작할 수 있기 때문에 CPU는 8253칩을 제어한 뒤 다른 일들을 처리할 수 있다. 따라서 8253은 리얼타임(Real Time) 클럭으로 동작한다. 그러나 8253이 한 주기동안 데이터를 받아 들여 프로그램할 수 있는 최대의 길이는 약1 / 12초이다.

또한 이 8253 타이머칩은 시간을 분이나 시로 변환시키는데는 좀 색다른 방법이 요구되고 있기 때문에 타이머칩의 채널0에 들어온 펄스는 65535로 카운트되고, 8259 인터럽트칩에서 인터럽트를 실행한 뒤 BIOS 데이터영역에서 변수로 기록된다. **그림 1**에 그 과정을 보였다.

그림 1과 같은 과정을 통해 BIOS 데이터영역에서 기록된 변수들은 보통 시간(Time-of-Day)으로서 표현된다. 채널0에서 출력된 펄스는 실행중인 CPU 프로그램을 잠시 정지시키고 초당 18.2번의 하드웨어적인 인터럽트(그림 1에서 타이머 인터럽트)를 일으켜 BIOS 데이터영역의 카운트를 증가시킨다.

보통 0카운터는 밤 12 : 00시를 나타낸다. 즉 카운터가 24시에 도달할 때 0으로 리세트된다. 그리고 다른 시간들은 각 초당 18.2로 카운트를 나눠 계산된다. 시간 카운트는 대부분 타이밍동작에 관여한다.

8253/8254 타이머칩

8253 타이머칩(AT에서는 8254)에 있는 3개의 채널은 각각 3개의 레지스터로 구성된다. 이 3개의 레지스터는 각각의 채널0~2에 대해 40H~42H로 번호가 지정된 싱글포트(Single Port)를 통해 액세스된다.

포트는 채널을 통하여 데이터를 주고 받는 8비트 I / O레지스터의 도입선이다. 채널이 프로그램되었을 때 2바이트 값이 포트를 통해 액세스되는데 이때 낮은 바이트를 우선적으로 액세스한다. 액세스된 번호는 번호를 유지하는 16비트 래치 레지스터로 패스(Path) 되고, 그 16비트 래치 레지스터로부터 카피된 번호는 16비트 카운트레지스터내에 위치한다.

카운트레지스터내의 번호는 시스템클럭에서 채널로 펄스가입력될 때마다 1씩 감소한다. 이렇게 감소된 번호가 0에 도달할 때 채널은 출력펄스를 발생시킨 뒤 또다시 번호를 유지하는 래치 레지스터내에서 새로운 번호를 카피하여 카운트레지스터로 이동한다. 카운트레지스터내의 번호가 감소할 때마다 비트(Beat)는 점점 빨라진다.

또한 이 3개의 채널들은 능동적이기 때문에 CPU가 그 채널들을 자유자재로 끄거나 켜거나 할 수 없으며, 어떤 카운터레지스터의 현재 값들이라도 그 카운트를 혼란시키지 않고 임의의 시간에 읽어들여진다.

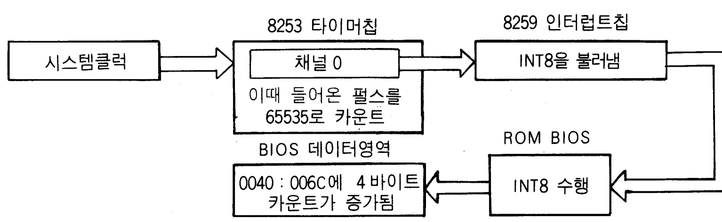
각 채널은 2개의 들어가는 라인과 1개의 나오는 라인을 갖고 있다. 이 3개의 라인을 통해 출입하는 신호의 목적지(Destination)는 IBM-PC의 형태에 따라 변한다.

그러면 채널 0~2에 대해 살펴보자.

●채널0

이 채널0는 시스템 시간클럭으로 사용된다. 그리고 이 채널은 스타트업(Start Up)할 때 BIOS에 의해 1초에 18.2번의 펄스를 발생시키도록 세트된다. 이러한 펄스들의 4바이트 카운트는

〔그림 1〕 BIOS 시간카운트를 새롭게 하는 과정



메모리내에 0040 : 006C(최소 특정 바이트는 가장 낮은 위치에 있음)에 놓인다. 그리고 각 펄스는 타이머 인터럽트(INT8)를 발생시켜 그 카운트를 증가시킨다. 이 타이머 인터럽트는 하드웨어 인터럽트로서 CPU의 동작과는 관계없이 하드웨어 인터럽트가 가능한 한 계속 발생한다.

채널0의 출력라인은 디스크동작의 타이밍을 확실하게 카운트하기 위해 사용되기 때문에 만일 출력라인을 변경시키려면, 디스크가 액세스될 때마다 원래 읽은 값으로 분명히 복원시켜야만 한다.

●채널1

메모리의 리플레시(Refresh)를 제어하며, 이 기능은 거의 변경되지 않는다. 이 채널의 출력라인은 직접 메모리 액세스칩(Direct Memory Access Chip)에 연결되어 있으며, 펄스는 DMA칩으로 모든 RAM을 리플레시하도록 되어 있다. 채널1은 디스크동작이 완료된 후 카운트가 새롭게(Update)되도록 시간클럭의 간섭펄스들을 카운트하는데 사용된다.

●채널2

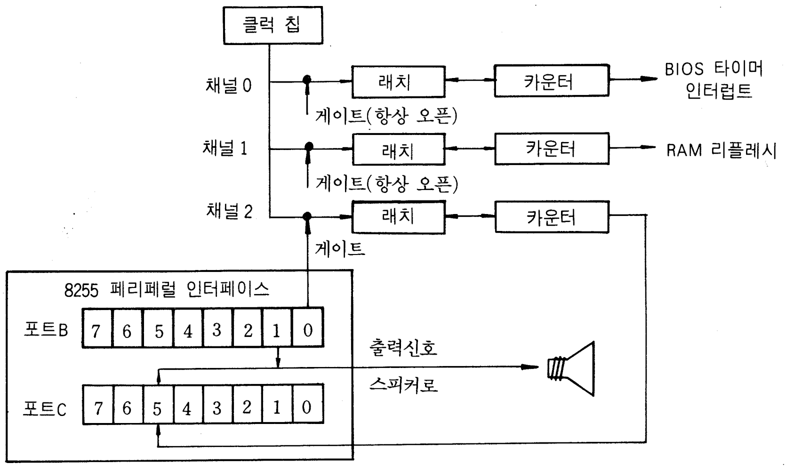
컴퓨터의 스피커에 연결되어 있으며, 단순한 구형파 신호를 발생시켜 스피커로 소리가 출력되도록 한다. 프로그래머들은 다른 사람들보다 채널2를 더 제어하고 있다. 왜냐하면 채널2를 어떻게 제어하느냐에 따라 간단한 소리는 다른 프로그램동작과 동시에 발생하도록 만들 수 있으며, 좀더 복잡한 소리들은 CPU에 모든 용량을 집중(Full Attention)시켜 발생할 수도 있다.

또한 채널2는 스피커에 접속되지 않아도 타이밍동작을 카운트하는데 사용될 수 있다. 그러나 최종적으로 채널2의 출력라인은 컴퓨터의 스피커에 접속되지만 스피커는 8255페리페럴 인터페이스칩에서 특별한 세팅이 이루어지지 않는 한 소리나지 않는다.

각 채널로 들어가는 2개의 라인은 시스템 클럭칩으로부터 시스템 클럭신호를 공급하는 1개의 클럭라인과 클럭신호를 ON / OFF시키는 1개의 게이트라 부르는 라인으로 구성된다. 그 ON / OFF시키는 게이트는 채널0 및 1의 클럭신호에 대해 항상 열려있다. 그러나 채널2의 클럭신호에 대해서는 열리거나 닫힐 수 있다.

이러한 채널2의 특징으로 인해 특수한 주파수의 소리도 낼 수 있다. 또 게이트는 8255칩상의 한 레지스터인 포트어드레스 61H에서 최하위 비트인 1로 세팅함으로서 닫힐 수 있지만 그 비트를 0으로 바꾸면 그 게이트는 다시 열린다. 그리고 이 게이트는 채널2의 출력과 같이 61H에서 비트1로 세팅되면 스피커에 연결되기 때문에 소리를 내는데 사용된다(그림 2 참조).

〔그림 2〕 8253/8254 타이머칩 블록다이어그램



타이머칩은 타이밍동작에 직접 사용될 수 있지만, 이것은 거의 실용적이지 못하다. 왜냐하면 입력 클럭레이트가 1초에 1.19818백만번이고(시스템클럭이 더 빠르게 동작하는 AT에서조차 타이머칩은 1.19MHz신호를 수신한다), 16비트가 유지할 수 있는 가장 큰 번호는 65535이므로 그 번호를 18.2번의 클럭펄스 레이트로 나누기 때문에 데이터 수신이 가능한 펄스간의 가장 긴 주기는 대략 1 / 12초이다.

대부분의 타이밍동작은 BIOS의 시간카운트를 대신 사용한다. 여기서 한 주기는 시간값을 읽는 것으로 시간이 경과되지만, 주기동안 얼마나 많은 펄스들이 패스되었는가를 알아보기 위해 이전의 기준값과 비교한다.

하드웨어 설계자가 8253의 각 채널을 동작시키는데 있어서는 6모드가 제공된다. 프로그래머의 경우에는 보통 타이밍을 위한 채널0 또는 타이밍이나 소리를 위한 채널2와 모드3에 한하여 제공된다.

이 모드3에서는 일단 래치 레지스터가 번호를 받으면 카운터 레지스터로 즉시 카피하여 로드한다. 또 번호가 0이 되었을 때 래치는 순간적으로 카운터를 재로드하는 등 카운트의 반주기동안 출력라인은 "ON"이고, 나머지 반주기동안은 "OFF"이다. 그 결과로 소리를 내는 것과 카운트하는데 있어서 공통적인 부분은 구형파 형식이 유용하게 쓰인다는 점을 알 수 있다.

8비트 명령레지스터는 번호가 어떻게 채널로 로드되는가를 제어한다. 이 명령레지스터는 43H 포트어드레스에 놓인다. 명령레지스터는 어떤 채널이 어떤 모드로 프로그램되고, 래치의 바이트중 하나 또는 양쪽 모드가 번호를 수신할(보낼) 것인지를 지시하는 바이트로 주어진다. 명령레지스터는 또한 번호가 바이너리 데이터인지 BCD 데이터인지를 보여준다. 비트형식은 다

음과 같다.

```
비트 0    만일 0이면 바이너리 데이터이고,아니면
          BCD 데이터이다
    3-1   모드번호,0-5(000-101)
    5-4   동작의 종류:
           00=래치로 카운터값 설정
           01=하이 바이트만 읽고/쓴다
           10=로우 바이트만 읽고/쓴다
           11=하이 바이트를 읽고/쓴 뒤 로우
             바이트 읽고/쓴다
    7-6   프로그램된 채널의 수, 0-2(00-10)
```

다음은 8253칩을 프로그램 하기 위한 세가지 기본단계를 설명하고 있다. 일단 3단계가 완료되면 프로그램된 채널은 바로 새로 세팅되어 동작하기 시작한다.

1단계 : 채널, 읽기/쓰기상태, 운영모드, 그리고 수의 형태를 선택하는 비트 형식을 갖는 명령레지스터 43H에 한 바이트를 보낸다.

2단계 : 만일 채널 2 라면 포트어드레스 61H에서 비트 0과 1 을 세팅함으로서 클럭신호를 가능하게 한다.(레지스터의 비트 1이 1로 세트될 때, 채널2는 스피커를 구동한다. 타이밍으로 동작시키기 위해서는 비트1을 0으로 세트한다.)

3단계 : 0～65535번호에서 카운터를 계산하고, 카운터 값을 AX 레지스터에 놓은 뒤 채널의 I / O레지스터(40H ～ 42H) 에 하위바이트를 먼저 보내고 그 다음에 상위바이트를 보낸다.

8253의 3채널은 항상 동작상태에 있다. 따라서 프로그램이 완료되기 전에 8253 레지스터는 원래의 세팅상태로 복원되어야 한다. 특히, 만약 소리의 프로그램이 끝났을지라도 8253 레지스터의 복원은 진행중이다. 왜냐하면 DOS가 다른 프로그램을 제어하여 로드한 후라도 계속되기 때문이다. Ctrl-Break을 사용하여 루틴을 빠져 나갈때 8253의 3채널은 항상 동작한다는 사실을 염두에 두어야 한다.

● 어셈블리어의 경우

이 예제에서 채널0은 앞에서 설명했던 것처럼 BIOS에 의해 이루어진 세팅과는 다른 값으로 프로그램된다. 여기서 세팅을 변경하는 이유는 시간카운트가 1초에 18.2번보다 빠른 속도로 증가하기 위한 것으로, 그 속도는 정확한 실험측정을 하기 위해 초당 1000번 변한다. 이렇게 할 경우 래치값은 1192 (1193180 Clocks Per Second / 10000)가 되어야 한다.

그리고 디스크를 동작시키기 이전에 원래의 래치값은 채널0이 그 동작의 타이밍을 제어하기 때문에 채널0의 값으로 대치시켜야 한다. 이 값은 가능한 최대값이어야 함으로 채널로부터 인가된 펄스들 사이에 가장 최대값인 65535를 채널0에 입력하고 래치 레지스터에 0을 대치시킨다. (0은 즉시 65535로 카운트된다.) (리스트 1 참조).

시간의 세트 및 읽기

처음 컴퓨터의 전원을 ON시켰을 때 DOS는 컴퓨터 유저에게 시간을 입력하도록 지시한다. 이때 시간을 입력하면 이 값은 시간 카운트(가장 낮은 최소 특정바이트인 0040 : 006C에서 시작하는)를 갖는 4개의 바이트내에 놓이게 되며 이 값은 시간이 카운트되는 형태로 변환된다. 즉, 시간은 자정이 지난 18번째 초의 값을 재현하는데 카운트는 타이머 인터럽트에 의해 초당 18.2번 계속적으로 새롭게 바뀐다. 여기서 시간에 대해 연속적인 요구가 있을 경우 시간 카운트의 현재 값은 1 / 18초인 시간카운트에 익숙한 시 / 분 / 초 형식으로 다시 변환한다. 만일 스타트업에서 입력된 값이 없다면, 카운트는 마치 자정인 것처럼 0으로 세트되며, 리얼 타임 클럭이 내장된 컴퓨터일 경우에는 자동적으로 시간 카운트를 세트한다.

●베이직의 경우

TIME$는 0에서 23까지 카운트되는 시간에 대해 자정부터 시작하여 hh : mm : ss 형태로 스트링(String)처럼 시간을 세트하거나 복원한다. 그러면 5 : 10 PM의 경우를 살펴보자.

```
100 TIME = "17:10:00"   '시간설정
110 PRINT TIMES          '시간호출
```

TIME$은 스트링으로 되돌아갈 수 있기 때문에 스트링기능은 MID$, LEFT$, 그리고 RIGHT$와 같이 시간을 읽을 수 있는 임의의 특수부분을 선택하는데 필요하다. 예를 들어 17 : 10 : 00에서 5 : 00까지 시간을 변환하기 위해서는 시(Hour)를 나타내는 스트링으로부터 캐릭터를 제거해야 하고 그 캐릭터들을 숫자의 형태로 변환시켜(VAL을 사용), 12를 뺀뒤 그 결과를

〔리스트 1〕

```
;---I/O레지스터 셋업:
 COMMAND_REG  EQU  43H          ;코멘드레지스터의 어드레스 설정
 CHANNEL_0    EQU  40H          ;채널0의 어드레스 설정
              MOV  AL,00110110B ;채널2,2바이트 카운터,.
                                ;모드3,바이너리번호에 대한 비트패턴
           COMMAND_REG,AL       ;코멘드레지스터로 바이트 출력
;---래치로 카운터 출력:
              MOV  AX,1193      ;100pulses/sec의 카운터
              OUT  CHANNEL_2,AL ;LSB 출력
              MOV  AL,AH        ;AL을 출력하기 위해 MSB시프트
              OUT  CHANNEL_2,AL ;MSB 출력
```

스트링 형식으로 다시 변환한다. (리스트 2 참조).

●어셈블리어의 경우

DOS는 시간카운트에서 시 / 분 / 초까지 요구되는 변환을 하면서 시간을 읽어 들이거나 세트하는 인터럽트를 제공한다. 시간은 1 / 100초의 정확성으로 세트되지만 시간카운트는 이 레이트(Rate)의 5번째에서 새롭게(Update) 세트되기 때문에 초당 100번째 읽어들인 시간은 근사적인 것이다.

INT 21H에서 2CH 기능은 시간을 복원하며, 2DH기능은 시간을 세트한다. 두 경우에는 CH는 시(0~23, 여기서 0=자정)를 유지하고, CL은 분(0~59)을 유지하며, DH는 초(0~59)를 유지한다. 그리고 DL은 "100초"(0~99)를 유지한다.

더우기 2CH기능이 시간을 출력했을 때 AL플래그는 한 주(Week)의 요일(0=일요일)번호를 유지한다. 날짜는 날짜가 세트되어 있어야 정확하게 출력된다. DOS는 날짜로 한주의 요일을 계산한다. 2DH 기능이 시간을 세트할 때 AL플래그는 시간에 대해 입력된 값이 유효한가를 체크한다(0=유효, FF=무효).

```
;---시간 세트:
    MOV  CH,HOURS          ;시간을 입력
    MOV  CL,MINUTES        ;
    MOV  DH,SECONDS        ;
    MOV  DL,HUNDREDTHS     ;
    MOV  AH,2DH            ;시간설정을 위한 기능
                            번호
    INT  21H               ;시간설정 실행
    CMP  AH,0FFH           ;시간이 유효한지 정확히
                            체크
    JE   ERROR             ;만일 시간이 정확치
                            않으면 에러루틴으로 점프
;---시간 복원:
    MOV  AH,2CH            ;시간설정을 위한 기능
                            번호
    INT  21H               ;시간설정 실행
    MOV  DAT_OF_WEEK,AH    ;AH로 부터 주의 요일
                            받음
```

만약 특수 응용을 위해 8253 칩의 채널1의 펄스속도를 변경시키고자 한다면, 독자 자신의 루틴으로 시간카운트를 디코드해야 한다. BIOS는 1,573,000펄스후에 카운트를 0으로 되돌리는데, 이 변환은 단지 타이머 인터럽트를 릴라이팅함으로서 가능하다.

따라서 1 / 100초에 해당되는 클럭은 어떤 특수 프로그래밍이 없다면 24시간에 대해 동작할 수 없다. 00 : 40 : 0070에서의 바이트는 스타트업할 때 0으로 세트되며, 클럭이 변경되었을 때 1로 증가한다. 그러나 그 이상 증가하지는 않는다.

날짜의 세트와 리드

컴퓨터가 켜졌을 때 DOS는 유저로 하여금 현재의 날짜 및 시간을 입력할 수 있도록 프롬프트한다. 그러나 날짜는 COMMAND.COM의 변수에 위치하고 있다. 각각 한달의 해당되는 날짜와 일년중 해당되는 달의 번호, 그리고 해당되는 연도의 번호를 0에서부터 카운트된 것을 갖는 3개의 연속된 바이트로 포맷된다. 여기서 0은 1980년을 가리킨다.

시간 카운트와는 달리 날짜의 메모리위치는 DOS버전과 메모리내의 COMMAND. COM의 위치에 따라 변한다. 따라서 날짜는 직접 패치(Fetch)되기 보다는 항상 BASIC이나 DOS에 이미 마련된 유틸리티들을 이용하여 액세스되어야 한다.

클럭이 내장된 경우 특수 S/W(보통 AUTOEXEC. BAT화일을 통해 스타트업에서 즉시 실행되는)의 도움으로 시간 및 날짜를 자동적으로 세트한다. 어떻게 클럭 캘린더칩이 액세스하는가는 뒤에서 설명하겠다. 그리고 BIOS 시간 카운트가 24시후에도 계속될 때 그에 따라 DOS는 날짜를 조정한다.

●베이직의 경우

DATA$는 날짜를 mm-dd-yyyy 형태로 스트링하면서 세트하거나 복원한다. 이때 대시(-)대신 슬래시(/)를 사용해도 된다. 연도의 처음 2개의 디지트(Digit) yy는 생략될 수도 있기 때문에 날짜는 mm-dd-yy의 형태로 표현이 가능한데 대부분 후자의 형태를 사용한다.

```
100 DATA$="10/31/84"          '날짜 설정
110 PRINT DATA$               '날짜 출력

...스크린에 디스플레이: 10-31-84
```

●어셈블리의 경우

DOS인터럽트번호21H의 24H와 2BH 기능번호는 날짜를 출력하거나 세트한다. 날짜를 얻기 위해서는 AH내에 24H를 놓고 인터럽트를 수행한다. 그리고 되돌아오면 CX는 1980~2099(이것은 1980년부터 오프셋(Offset)된 날짜이다)년을 0에서 119까지의 번호로서 유지한다. 여기서도 월(Month)의 번호는 DH로 유지하고, 일(Day)의 번호는 DL로 유지한다.

〔리스트 2〕

```
100 T$=TIME$                               'T$에 TIME$스트링을 설정
110 HOUR$=LEFT$(T$,2)                      'T$의 2번째 왼쪽 캐릭터를 시간으로 출력
120 MINUTES$=MID$(T$,4,2)                  '2개의 캐릭터를 분으로 출력
130 NEWHOUR=VAL(HOUR$)                     'HOUR$를 숫자로 변환
140 IF NEWHOUR>12 THEN NEWHOUR=NEWHOUR-12  '변환된 숫자에서 12를 뺌
140 NEWTIME$=STR$(NEWHOUR)                 '스트링형태로 새로운 값을 다시 변환
150 NEWTIMES=NEWHOUR$+":"+MINUTESS         '시,:,분의 형태로 새로운 시간설정
```

```
MOV AH,2AH  ;날짜를 복원하기 위해 AH나에 2AH를 놓음
INT 21H     ;날짜를 출력하기 위해 인터럽트 수행
MOV DAY,DL  ;DL에서 일(Day)을 받음
MOV MONTH,DH ;DH에서 월을 받음
ADD CX,1980 ;입력날짜에 기존에 세팅된 날짜를 더함
MOV YEAR,CX ;만약 CX가 5라면,5+1980=1985로 세팅
```

날짜를 세트하기 위해서는 같은 레지스터에 일, 월, 년을 놓고 기능번호 2BH를 수행한다. 만일 날짜에 대한 값이 유효하다면, AL은 FF로 보내고, 그렇지 않으면 0으로 보낸다.

```
MOV DL,DAY    ;DL에 일의 데이터를 놓음
MOV DH,MONTH  ;DH에 월의 데이터를 놓음
MOV CX,YEAR   ;CX에 년의 데이터를 놓음
               (예를 들면 1985)
SUB CX,1980   ;1980으로 오프셋된 기존 데이터를
               CX에 더함
MOV AH,2BH    ;세트된 데이터를 기능번호로
INT 21H       ;데이터 실행
CMP AH,0FFH   ;0FFH를 기준으로 제대로
               동작하는지 체크
JE ERROR      ;정의된 범위외의 데이터이면
               에러루틴으로 돌아감
```

리얼타임클럭의 세트와 리드

리얼타임클럭은 말 그대로 실제 시간을 그대로 전달할 수 있는 클럭으로서 다른 컴퓨터의 운영에도 방해받지 않고 시간을 카운트할 수 있는 독립적인 프로세서를 갖고 있다. 또한 컴퓨터가 꺼졌을 때 클럭이 계속 동작할 수 있도록 배터리로 전력공급을 한다.

프로그램은 리얼타임클럭을 읽고 세트할 수 있다. 일반적으로 보조 소프트웨어는 BIOS 시간카운트와 리얼타임클럭의 현재 세팅을 반영하기 위한 DOS의 날짜 변수를 세트하게 된다. 그러나 프로그램은 이러한 DOS의 변수들을 사용하기전에 이 값이 현재 값인가를 체크하게 되고, 만일 어떤 모순이 발견되면 그 문제를 즉시 세트할 수 있다.

클럭으로 여러시간 및 날짜를 세팅하는 것은 일련의 포트어드레스를 통해 이루어진다. IBM-PC에서 사용이 가능한 다기능 보드의 대부분은 리얼타임클럭을 갖고 있지만 유감스럽게도 칩이나 포트어드레스에 대한 표준이 없다.

AT는 모토롤라 MC146818 칩을 근간으로 한 리얼타임클럭을 장착하고 있으며, 칩상의 레지스터에 그 시스템을 위한 구조 데이터를 할당하고 있다. 레지스터들은 우선 포트어드레스 70H에 레지스터 번호를 보내고 71H에서 레지스터 값을 읽어 들임으로서 액세스된다. 클럭에 관련된 레지스터들은 다음 표 1과 같다.

표 1의 4개의 상태 레지스터에 비트들은 여러가지 기능을 수행하며, 그중에서도 프로그래머들이 많은 관심을 갖고 있는 것들은 표 2와 같다.

〔표 1〕 클럭에 관련된 레지스터

레지스터 번호	기 능
00H	초
01	초 알람
02	분
03	분 알람
04	시
05	시 알람
06	주(Week)의 날짜
07	월(Month)의 날짜
08	월
09	년
0A	상태 레지스터 A
0B	상태 레지스터 B
0C	상태 레지스터 C
0D	상태 레지스터 D

AT의 실시간 클럭은 하드웨어 인터럽트 IRQ 8을 불러낼 수 있다. 프로그램은 특별한 시간에서 수행되기를 원하는 어떤 루틴에 대하여 이러한 IRQ 8인터럽트에 대한 벡터를 지시하기도 한다. 이때 벡터는 4AH를 사용한다.

이러한 방법에 의한 실시간동작에는 뒤에서 설명할 실시간동작보다는 적은 프로세싱 오버헤드를 필요로 한다. 그리고 IRQ 8 인터럽트는 세가지 방법으로 불러낼 수 있는데, 이러한 방법은 모두 스타트업 단계에서는 불가능하다. 주기적인 인터럽트는 규칙적인 주기로 발생하며, 주기는 거의 1ms로 초기 설정된다. 알람 인터럽트는 3개의 알람에 관련된 레지스터의 세팅이 그에 대응하는 타이밍 레지스터들과 매칭될 때 발생한다. 새롭게 하는 작업이 완료된(Update-ended) 인터럽트는 칩상의 레지스터 세팅을 새롭게 한 후 발생한다.

INT 1AH는 실시간클럭을 세트하고 읽기 위해 AT의 BIOS에서 확장된다. 그러나 2개의 십진 디지트이상의 숫자는 읽을 수 없으므로 시간을 나타내는 숫자는 BCD(Binary Coded Decimal)로 주어진다. 여기서 BCD로 표현된 하나의 숫자는 4비트로 구성된 바이너리코드로 표현할 수 있기 때문에 이 BCD로 표현된 시간을 2진수의 형태로 다시 표현할 경우 ASCII코드로 바꾸는데 있어서 훨씬 수월하다.

모든 IBM기종에서는 INT 1AH의 기능 0과 1이 BIOS 시간카운트를 읽고 세트한다. AT의 실시간 클럭을 서

〔표 2〕 상태 레지스터

레지스터	비트	기능
레지스터 A	비트 7	1=진행하면서 시간을 새롭게(읽기전에 0까지 기다림)
레지스터 B	비트 6	1=주기적인 인터럽트 가능
	비트 5	1=알람 인터럽트 가능
	비트 4	1=새롭게 하는 작업이 완료된 인터럽트 가능
	비트 1	1=24에 의해 카운트된 시간 0=12에 의해 카운트된 시간
	비트 0	1=일광절약시간(Daylight Savings Time)이 가능

비스하기 위한 6가지의 새로운 기능들은 표 3과 같다.

알람세팅은 세팅이 형성되는 시간부터 옵션이 형성되는데 최대 주기는 23:59:59이다. 앞에서 설명한 것처럼 인터럽트 벡터 4AH는 알람루틴으로 지시되어야 한다. 만일 축전지가 모두 소모되어 클럭이 동작을 안한다면 기능 2, 4, 6은 캐리 플래그(Carry Flag)를 세트한다.

프로그램운용에 있어서의 지연

프로그램운용이 빈 루프에 의해 지연됐을 때 적당한 기간동안 루프를 테스트한 뒤 다시 테스트하는데 상당한 양의 프로그래밍시간이 소모될 수 있다. 현재 직면한 프로그램에서 이 빈 루프의 길이를 알아내 프로그래밍시간을 단축할 수 있는 방법을 찾는다 해도 계속해서 주어지는 새로운 프로그램에는 적용할 수 없을 것이다.

루프의 속도는 사용된 컴파일러의 속도에 따라 변한다.(또는 베이직에서 루프속도는 프로그램이 컴파일되었는지의 여부에 따라 결정된다.) 그리고 현재의 CPU속도에 한계를 느끼게 하는 AT와 여러 IBM호환기종이 출현하고 있기 때문에 어셈블리언어 루프조차도 계속적으로 변할가능성이 높다. 따라서 정확하게 클럭된 지연프로그램을 만들어 놓는 것이 좋은데, BIOS시간카운트의 18.2번/초의 펄스속도가 가장 적당하다.

세트시간을 지연시키기 위해서 프로그램은 시간카운트내의 몇개의 펄스가 그 지연시키고자하는 기간과 일치하는지 계산하고 그 값을 카운트의 현재 읽혀진 값에 더한다. 그리고 프로그램은 카운트를 읽어 기대값과 비교한다. 두 값이 같을 때는 지연이 이루어지고 프로그램은 계속 진행한다. 시간 카운트를 유지하는 4바이트는 0040:006C에서 시작한다(예전과 같이 최소 특정 바이트는 메모리에서 가장 낮다).

●베이직의 경우

베이직에서는 주파수에 대해 32767값을 갖는 SOUND문을 사용한다. 이 SOUND문을 사용할 경우에는 전혀 소리가 나지 않는다. 이러한 소리가 나지 않는 상태는 지정한 지연기간의 시간펄스동안 계속 유지된다. 5초간 지연할 경우에는 91개의 펄스(5×18.2)동안 소리가 나지 않게 된다. 5초동안 프로그램 지연에 대한 내용은 다음과 같다.

```
100 SOUND 32767,91   '5초동안 프로그램 지연
```

그리고 시간카운트를 직접 읽는 경우에는 다음과 같다.

```
100 DEF SEG=0 '메모리 최하단에서 세그먼트 세트
110 LOWBYTE=PEEK(&H46C) '최하위 바이트 출력
120 NEXTBYTE=PEEK(&H46D) '그 다음 바이트 출력
130 LOWCOUNT=NEXTBYTE*256+LOWBYTE '두개의 하위
                                  바이트 값을 결합
```

●어셈블리어의 경우

INT 1AH의 기능 0을 사용하는 BIOS 시간카운트를 읽고 그 값에다 1/18초 번째 펄스의 해당번호를 더한다. 그리고 해당번호에 대한 현재 값을 테스트할 때마다 시간카운트를 재차 읽어낼 수 있도록 한다. 그래서 해당번호의 값과 시간카운트 값이 일치하면 지연은 끝난다. INT 1AH는 DX내에 있는 두개의 하위바이트를 되돌리고(대부분의 지연이 카운트될 수 있는 범위내에서), 이미 되돌려진 DX내의 두 상위바이트는 무시된다. 다음의 예는 5초의 지연에 91펄스가 해당된 것이다.

```
;---BIOS 값을 출력하고 지연 값을 더함:
        MOV AH,0     ;읽기위한 기능번호
        INT 1AH      ;시간카운트를 실행
```

〔표 3〕 AT의 실시간 클럭을 서비스하기 위한 6가지 새로운 기능

기능2 : 실시간 클럭으로 부터 시간 읽기
복원상태 : CH=BCD에 시간 CL=BCD에 분 DH=BCD에 초
기능3 : 실시간 클럭에서 시간 세트
입력상태 : CH=BCD에 시간 CL=BCD에 분 DH=BCD에 초 DL=일광절약시간이거나, 아니면 1
기능4 : 실시간 클럭으로 부터 데이터 읽기
복원상태 : CH=BCD에 1세기(19 또는 20) CL=BCD에 년도(1980으로 오프셋) DH=BCD에 월 DL=BCD에 월의 날짜
기능5 : 실시간 클럭에서 데이터 세트
입력상태 : CH=BCD에 1세기(19 또는 20) CL=BCD에 년도(1980으로 오프셋) DH=BCD에 월 DL=BCD에 월의 날짜
기능6 : 실시간 클럭에서 알람을 세트
입력상태 : CH=BCD에 시 CL=BCD에 분 DH=BCD에 초
기능7 : 알람을 리세트(입력 레지스터 없음)

```
         ADD  DX,91   ;5초 더하고 하위워드만큼 지연
         MOV  BX,DX   ;BX에 지연완료값 저장
;---BIOS시간값 체킹 유지:
REPEAT:  INT  1AH     ;시간을 다시 읽어 출력
         CMP  DX,BX   ;지연값을 읽어 비교
         JNE  REPEAT  ;같지 않으면 REPEAT로 점프
                      ;아니면 지연이 완료
```

시간프로그램 운용

프로그램의 시간운용은 사람이 하는 것처럼 정확하다. 즉, 이것은 시스템에 시간카운트의 초기에 읽은 값을 설정한 다음 그 초기설정 값과 연속적으로 읽어지는 값을 비교한다. 그 읽는 형식은 시/분/초로 설정되지만 카운팅 시스템은 십진법으로 표기하지 않기 때문에 이 읽은 두 값사이의 차를 계산하는 것은 무척 번거롭다. BIOS 시간카운트는 직접 읽어들이는 것보다 1/18초내의 경과된 기간을 측정하여 hh:mm :ss 형식으로 바꿔 읽는다.

●베이직의 경우

베이직에서는 0040:006C의 메모리 위치에서 직접 BIOS 카운트를 읽는다. 경과된 시간을 쉽게 계산하기 위해 번호를 65520으로 나누면 분당 1092이고, 초당 18.2가 경과된 시간이다.

```
100 GOSUB 500      ;시간카운트 출력
110 START=TOTAL    ;START에 초기 카운트값 저장
.
(지정된 시간만 움직임)

.
300 GOSUB 500 ;마지막 시간카운트 출력
310 TOTAL=TOTAL-START ;경과된 펄스 계산
320 HOURS=FIX(TOTAL/65520) ;시간에 대한 번호 계산
330 TOTAL=TOTAL-HOURS*65520 ;TOTAL에서 시간 감산
340 MINUTES=FIX(TOTAL/1092) ;분에 대한 번호 계산
350 TOTAL=TOTAL-MINUTES*1092 ;TOTAL에서 분 감산
360 SECONDS=FIX(TOTAL/18.2) ;초에 대한 번호 계산
370 PRINT HOURS,MINUTES,SECONDS ;결과
380 END
.
.
```

```
500 DEF SEG=0              ;시간을 읽기 위한 서브루틴
510 A=PEEK(&H46C)          ;최하위바이트 설정
520 B=PEEK(&H46D)          ;두번째 하위바이트 설정
530 C=PEEK(&H46E)          ;세번째 하위바이트 설정
540 TOTAL=A+B*256+C*65535  ;TOTAL에 카운트 기록
550 RENTURN                ;처음부터 다시
```

베이직에서 타이머기능은 시간카운트가 0으로 세트된 상태를 유지하면 패스되어온 초의 번호를 다시 되돌린다. 보통 이 초의 번호는 컴퓨터가 부트된 후의 것이 될 것이다. 만일 시간이 시스템의 스타트업상태에서 정확히 세트되었다면 타이머는 자정이후 패스된 초들의 번호를 다시 되돌린다. 따라서 간단히 N=TIMER를 쓴다. 여기서 N은 패스된 초의 번호이다.

●어셈블리어의 경우

INT 1AH는 시간카운트를 세트 (AH=1) 하고 복원하는 (AH=0) 두가지 기능을 갖고 있다. 카운트를 읽기 위해서는 간단히 AH에서 0으로 인터럽트를 수행한다. 되돌아가는 동작으로 CX:DX는 CX내의 가장 중요한 워드를 사용해 카운트한다. 카운트가 세트된 상태를 계속 유지하여 24시간을 패스하지 않는다면 AL은 0으로 세트된다. 카운트를 세트하기 위해서는 똑같은 레지스터에 2개의 워드를 놓고 AH를 1로 세트한다. 다음의 예는 1시간한도내에서 경과된 시간을 측정한 것이다. 단 여기서 카운터의 기본 2바이트는 고려될 필요가 있는데 이를 고려할 경우 초기 읽은 값이 두번째 읽은 것보다 높으면 "전도(Turnover)"조건이 분명히 허용된다. (리스트3 참조).

리얼타임 동작의 제어

리얼타임동작에서 프로그램은 가능한한 빨리 명령들을 유출하기 보다는 시간내에 지정된 지점에서 그 명령을 유출한다. 이러한 기법은 보통 로보틱스(Robotics)에 관련되지만 그외에도 다양한 용도에 적용된다.

리얼타임동작을 위한 하나의 접근방

〔리스트 3〕

```
;---데이터 세그먼트 :
  OLDCOUNT   DW  0             ;시간카운트 초기화 유지
;---시간의 초기 읽기:
           MOV  AH,0           ;기능번호 세트
           INT  1AH            ;카운트 실행(DX의 하위워드)
           MOV  OLDCOUNT,DX    ;초기 카운트 저장
;---나중에 경과된 시간 계산:
            MOV  AH,0               ;기능번호 세트
            INT  1AH                ;카운트 실행
            MOV  BX,OLDCOUNT        ;첫번째 읽은 값을 복원
            CMP  BX,DX              ;전도(Turnover)조건 체크
            JG   ADJUST             ;만일 전도되면 조정루틴으로 점프
            SUB  DX,BX              ;아니면 다른 점을 찾음(=경과된 펄스)
            JMP  SHORT FIGURE_TIME  ;시간계산 위해 조정을 통하여
                                                점프
;---전도에 의한 조정:
  조정:     MOV  CX,0FFFFH ;CX에 가장 큰 번호를 놓음
           SUB  CX,BX     ;첫번째 읽은 값은 감산
           ADD  CX,DX     ;두번째 읽은 값은 가산
           MOV  DX,CX     ;DX에 경과된 시간 남김
;---시간계산루틴 시작:
  FIGURE_TIME:            ;현재 초당 18.2에 시간에 의해 DX를 나눔
```

법으로 리얼타임명령간에서 거의 수행되지 않는 프로그램은 쉬고 있는 상태로 하고 단지 이 프로그램은 언제 시간이 동작될 것인가를 감지하기 위해 BIOS 시간카운트만을 체크하게 한다.

두번째 접근방법은 좀 어렵다. 이 기법은 프로그램이 일정하게 움직일 때 사용되지만, 어떤 작업을 수행하기 위하여 특정 시간에 그의 동작을 인터럽트한다. 확장작업을 수행할 경우에는 초당 18.2번 수행되는 타이머 인터럽트에 의해 이루어진다. 인터럽트가 발생할 때마다 그 확장은 시간카운트의 새로운 값을 체크하고 만일 리얼타임 동작을 시작하는 점이 카운트 값과 일치된다면, 루틴은 그 동작을 시작한다. **그림 3**에 이 과정을 보인다.

●베이직의 경우

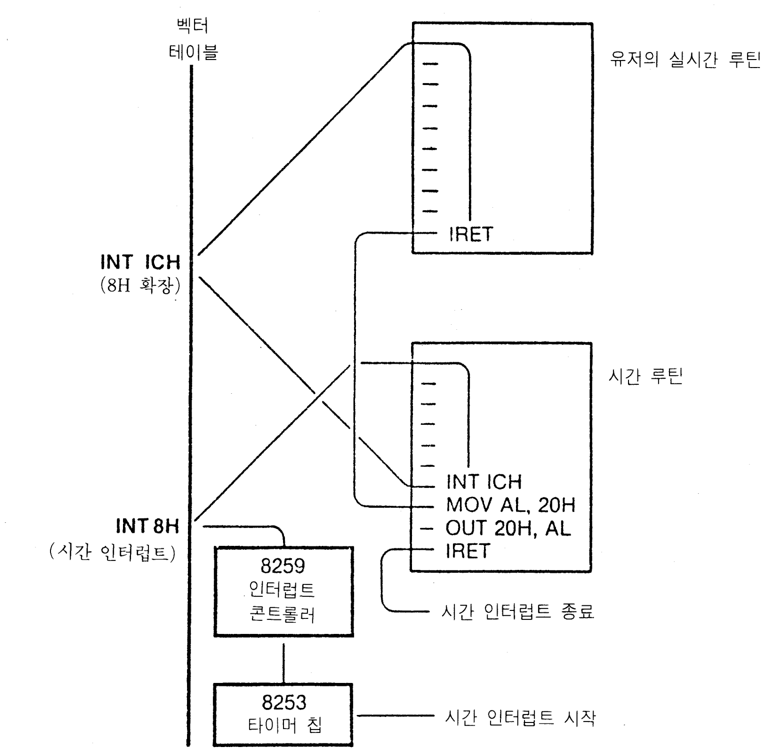
베이직은 리얼타임 동작에 대해 ON TIMER (n) GOSUB문으로 원시적인 제어를 한다. 프로그램이 이 문을 만났을 때 n에 의해 주어지는 초의 번호를 카운트하기 시작한다. 그동안 프로그램 동작은 계속된다. n초가 지났을 때 프로그램은 특정 라인 번호에서 시작하는 서브루틴으로 점프하여 그 서브루틴을 수행한 뒤 원래의 지점으로 되돌아간다. 따라서 카운팅은 0에서 다시 시작하고 서브루틴은 n초후에 다시 호출된다.

ON TIMER는 TIMER ON문으로 쓸모 있을 때까지는 그 기능을 발휘하지 않는다. 이 ON TIMER문에는 TIMER OFF문은 쓸모없다. 또한 타이밍은 계속되지만, 서브루틴으로의 전송이 지연되는 경우에는 TIMER STOP문을 사용하여 일단 타이밍을 중지시킨다. 이러한 경우에는 n초 지난 것으로 기록되며, 프로그램은 또 다른 TIMER ON문을 만나면 곧 바로 서브루틴으로 점프한다.

이러한 동작이 계속 반복되기 때문에 ON TIMER문은 특히 스크린상에서 클럭을 나타내는데 유용하다.

```
100  ON TIMER(60) GOSUB 500        '60초마다 클럭이
                                    변함
110  TIMER ON                      '타이머 가능
.
.
500  LOCATE 1,35:PRINT"TIME: ";LEFT$,5) '커서의
                                    위치를 정하고, 시간출력
510  RETURN
```

〔그림 3〕



●어셈블리어의 경우

BIOS는 자신을 위한 루틴을 제공받지 않는 한 아무 일도 하지 않는 특별한 "더미((Dummy)" 인터럽트(INT 1CH)를 갖고 있다. 스타트업에서 인터럽트에 대한 벡터는 IRET(인터럽트 되돌림(Return)) 명령으로 지시하며, 그 인터럽트가 호출되었을 때 간단히 되돌리면 된다.

INT 1CH에 비해 특별한 것은 인터럽트가 시간카운트를 새롭게 한 후에 BIOS타이머 인터럽트에 의해 호출된다는 것이다. 말하자면 그것은 초당 18.2번 자동으로 발생하는 하드웨어 인터럽트이다. 또한 이러한 인터럽트에 대한 벡터는 프로그램에서 프로시듀어(Procedure)를 지시하도록 바꿀 수 있다. 그래서 그 프로시듀어는 초당 18.2번 호출된다.

다음의 예에서 루틴(보이지 않는)은 알람이 사라지기 전에 분(Minute)의 번호를 패스하기 위해 60까지의 값을 유저에게 요구한다. 분에 저장된 번호는 시간펄스와 같은 값을 제공하도록 1092와 곱한다. 펄스번호는 현재 읽어들인 시간의 하위워드에 더한뒤 알람 카운터로서 저장된다.

다음에 인터럽트 1CH에 대한 벡터는 알람이라 부르는 프로시듀어를 지시하도록 변경된다. 일단 변경된 벡터를 저장하고 알람은 즉시 매 1/18초마다 호출되기 시작한다. 알람이 호출되면 1AH 인터럽트를 통해 읽혀지는 현재 시간을 얻은 뒤, 비교하기 위한 알람카운터를 복원한다.

만일 두 값이 일치하면 루틴은 스피커가 발신음을 내는 것으로, "BEEP"라 부르는 프로시듀어를 호출한다. 만일 두 값이 같지 않으면 루틴은 간단하게 되돌려진다. 하드웨어 인터럽트(MOV AH, 20H / OUT 20H, AL)에 대한 일반적인 되돌림코드(Return Code)는 타이머 인터럽트에 의해 처리되기 때문에 필요없다. (리스트 4 참조)

타이머 칩에 의한 랜덤번호의 발생

일련의 랜덤번호를 발생시키는데에는 중요한 수학적인 궤변성이 필요하다. 그러나 간혹 프로그램은 어떤 특별한 순간에 단일 번호를 요구하는 경우가 있다. 이 경우에 랜덤번호는 타이머 칩의 채널로부터 현재의 값을 읽게함으로서 간단히 프로그램이 요구하는 단일 번호를 호출할 수 있다. 베이직은 랜덤렬(Random Series)을 계산하기 위해 시드(Seed)와 같은 값을 사용한다. 물론 샘플링 레이트는 그 자체가 랜덤하지 않기 때문에 타이머 세팅을 연속적으로 읽게하여도 일련의 랜덤번호를 얻을 수 없다.

●베이직의 경우

베이직 프로그램이 수행될 때마다 일련의 다른 랜덤번호들이 만들어지도록 하기 위해 TIMER문을 사용하여 리시드(Reseed)될 수 있는 랜덤한 번호발생기를 갖고 있다. (리스트 5 참조)

```
100 RANDOMIZE TIMER         ,자동적으로 리시드될 수
                                          있는 발생기
110 PRINT RND,RND,RND       ,3개의 랜덤번호 출력
...출력: .7122483    .469555052    .9132487
```

●어셈블리어의 경우

타이머 채널의 카운터레지스터는 주어진 번호(당분간 0으로 세트)로 반복하여 재로드되기 때문에 랜덤번호가 요구하는 범위와 일치하는 카운트를 선택한다. 따라서 하루의 랜덤시간 카운터로 23을 사용하는 것이 적합하다.

또한 이 카운트는 타이머칩의 채널 2(포트 42H)에 모드3을 사용하는 것이 제일 좋다. 우선 원하는 범위(아래의 예는 0000에서 9999까지의 랜덤 값을 주기위해 10000을 사용한다)에 카운터를 세트한다. 그 다음 랜덤번호에 대한 채널을 샘플링하기 위해서는 비

〔리스트 4〕

```
;---데이터 세그먼트:
        MINUTES     DW  0  ;알람이 울릴 때까지 분의 번호 유지
        ALARMCOUNT  DW  0  ;알람 세팅위한 시간 유지
;---BIOS 대기시간 값으로 알람카운트 세트
        CALL REQUEST_MINUTES ;유저로부터 소리날 때까지 분 호출
        MOV  AX,MINUTES      ;AX로 분번호 옮김
        MOV  BX,1092         ;분당 시간펄스의 번호 BX로 옮김
        MUL  BX              ;AX내의 현재 결과를 곱함
;현재 시간을 출력:
        MOV  AH,0            ;시간을 읽기 위한 기능번호
        INT  1AH             ;DX에 하위워드 카운트 실행
;두 값 더함:
        ADD  AX,DX           ;현재시간에 알람시간 더함
        MOV  ALARMCOUNT,AX   ;알람시간 설정
;---더미 인터럽트 변화:
        PUSH DS              ;데이터 세그먼트 저장
        MOV  AX,SEG ALARM    ;알람루틴의 세그먼트 출력
        MOV  DS,AX           ;DS에 세그먼트를 놓는다.
        MOV  DX,OFFSET ALARM ;알람루틴의 오프셋 출력
        MOV  AL,1CH          ;변화시키기 위한 인터럽트벡터 번호
        MOV  AH,25H          ;벡터를 변화시키는 DOS기능
        INT  21H             ;벡터 변화 실행
        POP  DS              ;데이터 세그먼트 재저장
;
;---프로그램만 연속됨...새로운 인터럽트 초당 18.2번 발생
;
;---프로그램완료로 전의 인터럽트 벡터 복원:
        MOV  DX,0FF533H      ;INT 1CH에 대한 최초의 오프셋
        MOV  AX,0F000H       ;최초의 세그먼트
        MOV  DS,AX           ;DS에 세그먼트 놓음
        MOV  AL,1CH          ;변화를 위한 인터럽트벡터 번호
        MOV  AH,25H          ;벡터를 변화시키는 DOS기능
        INT  21H             ;최초의 벡터 재저장
                             ;등...
;---소리 알람 프로시듀어:
ALARM   PROC FAR             ;거리 먼 프로시듀어 만듦
        PUSH AX              ;변화된 레지스터 저장
        PUSH CX              ;
        PUSH DX              ;
;시간카운트 읽기:
        MOV  AH,0            ;시간을 읽기 위한 기능번호
        INT  1AH             ;DX에 하위워드 카운트실행
;알람시간에 상당하는 카운트 실행:
        MOV  CX,ALARMCOUNT   ;"Time's up"신호 변하도록 실행
        CMP  DX,CX           ;현재 읽은시간과 일치하는지 비교
        JNE  NOT_YET         ;아니면,루틴을 중지
;만일 두 카운트 일치하면 소리냄:
        CALL BEEP            ;Beep루틴은 보이지 않음
;그렇지 않으면 인터럽트로 부터 되돌아옴:
        POP  DX              ;변한 레지스터 재저장
        POP  CX              ;
        POP  AX              ;
        IRET                 ;인터럽트로 부터 되돌아옴
ALARM   ENDP                 ;프로시듀어 완료
```

〔리스트 5〕

```
;---포트어드레스 세트:
  COMMAND_REG   EQU  43H        ;세트 코멘드레지스터 어드레스
  CHANNEL_2     EQU  42H        ;채널2 어드레스 세트
                CALL SET_COUNT ;타이머 범위 세트

;---프로그램만 동작...그 다음 랜덤번호 요구:
                CALL GET_NUMBER ;램덤번호 호출

;---채널2 카운팅 시작
  SET_COUNT     PROC
                MOV  AL,10110110B ;채널2와 모드2 두바이트의 바이너리
                OUT  COMMAND_REG,AL;코멘드레지스터에 명령바이트 출력
                MOV  AX,10000     ;카운터 값
                OUT  CHANNEL_2,AL ;카운터의 하위바이트 출력
                MOV  AL,AH        ;AL에 하위 바이트 옮김
                OUT  CHANNEL_2,AL ;카운터의 하위바이트 출력
                RET
  SET_COUNT     ENOP
;---랜덤번호 출력
  READ_NUMBER   PROC
;---래치레지스터로 카운터 값 옮김
                MOV  AL,10000110B  ;래치에 코멘드레지스터 지시
                OUT  COMMAND_REG,AL ;명령 출력
;---카운터 값 읽음
                MOV  AL,10110110B  ;읽기/쓰기에 대한 요구
                OUT  COMMAND_REG,AL ;요구출력
                IN   AL,CHANNNEL_2 ;하위바이트 입력
                MOV  AH,AL         ;AL에 하위바이트 임시적으로 유지
                IN   AL,CHANNEL_2  ;상위바이트 입력
                CALL SET_COUNT     ;래치레지스터의 값 재저장
                SWAP AH,AL         ;상위와 하위바이트 교체
                RET                ;그리고 현재 랜덤번호는 AX에 있다
 READ_NUMBER    ENDP
```

트4와 5를 0으로 세팅함으로서 카운터 레지스터의 현재 값을 "래치"하도록 포트43H에서 타이머칩 명령레지스터를 지시한다. 래치레지스터에 이러한 전달은 진행중인 카운팅에 방해를 주지 않는다.

다음에 CPU가 래치레지스터로부터 읽을 수 있도록 명령레지스터의 비트4와 5를 1로 세트한다. 그러면 두개의 IN명령은 우선 하위바이트를 AL레지스터에 가져가고 다음에 상위바이트를 가져간다. 마지막으로 카운팅이 원하는 범위에 걸쳐 계속되도록 래치레지스터를 최초의 값으로 리세트한다. PC

스피디 3의 제작(2)

전윤호/서울대 제어계측공학과 대학원

초보자들이 범하기 쉬운 실수중의 하나는 하드웨어에만 비중을 두어 프로그램을 작성하고 디버깅할 시간적 여유를 두지 않는다는 것이다.

마이크로 로보트는 여러종류의 하드웨어와 소프트웨어가 복잡하게 맞물려 있으며 자체에 키보드나 디스플레이가 없기 때문에 디버깅이 무척 어렵다. 또 프로그램의 작성도 일반적인 프로그램과 달라서 여러가지의 인터럽트가 수시로 있고 자세 및 거리보정, 또 여러가지 경우에 대한 처리가 모두 다르기 때문에 명확한 구조를 갖도록 프로그램하기가 매우 어렵다. 따라서 최소한 하드웨어와 같은 비중의 시간이 할애되어야 한다.

Speedy-3의 모든 프로그램은 어셈블리 언어로 작성되었다. C 등의 고수준 언어를 사용할 수도 있었겠지만 어차피 많은 부분이 어셈블리로 작성되어 링크되어야 하고, 또 Speedy-2 때 비록 CPU는 다르지만 어셈블리로 한 경험이 있었기 때문에 계속 그렇게 하기로 했다.

프로그램중 많은 부분이 Z-80 코드를 대충 번역한 것이고, 일부는 테스트 코드가 제거되지 않은 채 들어 있는 것이 있을 뿐더러 계속된 디버깅에 의해 복잡하게 뜯어 고쳐진 부분들이 많기 때문에 프로그램 리스트를 읽고 이해하기는 상당히 어려울 것이다. 프로그램은 개발상의 편의를 위하여 몇 부분으로 나뉘어졌고 따로 어셈블된 후 나중에 링크되었으며 RS-232C에 의해 로보트의 RAM에 전송되어 실행되었다. 물론 마이크로 로보트에서 다시 PC로 데이터를 전송할 수도 있으므로 의심가는 변수의 값은 수시로 전송하도록 해서 디버깅했다. 간단한 모니터 프로그램을 작성해서 Speedy-3의 ROM에 실어 놓았으면 디버깅에 많은 도움을 받았겠으나 시간에 쫓기다 보니 그럴 여유가 없었다. 먼저, 프로그램의 전체 구조에 대해 설명하고 그 후 각 부분에 대해 설명하겠다.

새로운 대회규정에 적응

이전까지의 마이크로 로보트 프로그램은 비교적 단순했다.

확장 좌/우수법 등을 사용하여 일단 목표지점을 찾아내면 정지하고 그 후 조작자가 집어내어서 다시 출발시키면 지름길을 찾아서 간다는 것이다.

여기서 좌수법에 대해 잠시 설명하면, 왼쪽손을 미로벽에 대고 걸어가는 것과 같다고 하여 이런 이름이 붙었는데, 한마디로 왼쪽 우선의 알고리즘으로서 왼쪽에 벽이 없으면 왼쪽으로, 왼쪽이 막혀 있고 앞쪽이 트였으면 앞쪽으로, 오른쪽만 트여있다면 오른쪽으로, 세방향이 모두 막혀있다면 다시 돌아가는 것을 말한다. 이 방법은 시작점과 끝점이 미로의 가장자리에 있는 모든 미로에 대해 사용할 수 있으나, 마이크로 로보트 경연대회처럼 목표지점이 미로의 중앙에 위치한 경우에는 목표지점에 이를 수 없는 경우가 생긴다.

확장 좌수법은 이런 경우에 대비해서 이미 지난 길은 안간다는 식으로 몇가지의 제한 조건을 추가한 것을 말한다. 이에 대해서는 이미 많은 문헌에 소개가 되었으므로 자세한 것은 참조하기 바란다.

그런데 지난 6회 대회부터는 규정이 세계대회 규정에 따라 바뀌었기 때문에 로보트의 프로그램도 거기에 맞게 바뀌어야 했다. 바뀐 규정은 본지 88년 12월호에 소개되었는데 로보트 스스로 출발지점에 돌아와서 다시 출발해야 하고, 1차 주행의 시간도 일정비율 기록에 반영된다는 것이 그 요지이다.

새로운 규정은 미로탐색 알고리즘 보다도 최고 속도만을 중시하던 과거의 경향을 바꾸기 위해 만들어졌는데 한번이라도 정지해서 사람이 손을 대면 10초 삭감 규정을 적용받지 못하기 때문에 모험을 할 수가 없고, 좋은 성적을 거두려면 로보트가 완벽한 주행을 한번에 성공해야 하는 것이다.

필자의 개인적인 생각으로는 주로 학생들이 취미로 만드는 마이크로 로보트에게 이러한 완벽성을 요구하는 것은 지나치다고 생각되어 이를 반대했었지만 어쨌든 이 규정은 우리나라에서도 그대로 채택되었다.

이 규정은 채택되자마자 전 규정과의 차이가 드러나서 Turbo-4가 Speedy-3보다 빨랐으나 결국 준우승에 머무르고만 원인이 되었다. Turbo-4는 미로탐색의 알고리즘이 Speedy-3보다 낮은 수준이었고 특히 안전성이 부족해서 여러번의 주행을 실패한 후 마지막에야 2차 주행을 성공했던 것이다.

당연히 10초 삭감 규정의 적용을 받지 못했고 미로시간도 길어서 전체 경연 시간에서 매우 불리해진 것이다.

프로그램의 전체구조

Speedy-3는 바뀐 규정에 대응하여 다음과 같은 프로그램으로 구성되었다. 먼저, Speedy-2에서와 같은 방법으로 미로를 탐색하는데, 이때 루프 테스트가 적용되며 이에 대해서는 별도로 설명하겠다.

일단 목표지점을 찾으면 그때까지의 정보에 의해 지름길을 계산해서 원래의 지점으로 돌아갔다 온다. 그것은 미로시간이 작은 상태에서 일단 기록을 내기 위해서이다. 그런 후에 더 빠른 지름길이 있는지를 알기 위해 그전에 안가본 곳을 더 탐색한다. 모든 곳을 다 가본후에 (물론 루프 테스트에 의해 가볼 필요 없는 곳은 제외) 다시 원점으로 되돌아가서 목표지점까지 지름길로 간 후 돌아온다. 혹시 안가본 곳을 탐색하는 중에 주행이 중단될 경우를 고려해서 수동으로 2차 주행을 시킬 수도 있게 했다.

Speedy-3를 개발하면서 최단 시간 경로의 계산과 곡선회전을 실현시키려고 했었으나 마지막까지 시간부족으로 완성시키지 못했다. 그러나 그보다도 안정성의 확보에 노력을 기울인 결과 우승을 차지할 수 있었는데, 이는 앞으로 이 대회에서 입상을 노리는 사람들에게 하나의 교훈이 될 것이다.

즉, 속도를 올리기 전에 일단 확실히 주행을 할 수 있도록 해두는 것이 중요하다. 재미있는 것은 Speedy-3가 처음에 찾아낸 지름길이 나중에 찾아낸 것보다 오히려 더 빨랐다는 점이다. 나중에 찾아낸 지름길이 거리상으로는 더 짧았으나 회전이 많아서 그런 결과가 발생했는데 어쨌든 최단시간경로의 계산을 할 수 없었던 Speedy-3로서는 운이 좋았다고 밖에 할 수 없다. Turbo-4는 2차 주행을 한 후에 다시 곡선회전을 사용해서 3차 주행을 하도록 프로그램 되었었다.

필자가 미국 대회에서 보고온 MITEE-Ⅱ, Ⅲ도 일단 지름길을 알아낸 후에는 계속 출발점과 목표점을 왕복하면서 속도를 조금씩 더 올렸었는데 가능한 최고의 기록을 얻기 위해 채택할 만한 방법이다.

어느 경우에도 무엇보다 확실하게 완주를 하는 것이 가장 중요하다.

루프 테스트는 역대 여러 대회의 미로를 가지고 시뮬레이션 해보면 스텝수에서 5~20%정도 단축이 된다. 그러나 루프 테스트에 의해 안가본 곳은 대부분 복잡하게 되돌아 나와야 하는 곳이기 때문에 시간상으로는 그보다 차이가 클 것이다. 또 Speedy-3에서 사용한 루프 테스트는 절대적인 것으로서 틀림없이 가볼 필요가 없는 곳은 안가지만, 개정된 규정하에서는 혹시 그곳에 더 빠른 지름길이 있을 가능성이 있더라도 그안에 목표지점이 없다면 일단은 그곳으로 가는 것을 유보하는 것이 좋을 것이다.

또한, 초음파 센서는 몇 블록 더 앞을 볼 수 있기 때문에 직진 우선의 알고리즘을 사용하여 일단 속도를 내어 직진하는 것이 시간상 더 유리할 지도 모른다. 앞으로의 대회에서 우승을 목표로 하는 사람들은 이러한 점을 고려할 필요가 있을 것이다. 1차 주행시간이 짧아지면 물론 경연시간에서도 혜택을 받지만 그 보다도 벽에 부딪힐 확률이 그 만큼 줄어들어 빠르게 완주할 수 있다는 확신이 서면 건전지도 보다 더 가벼운 것을 사용할 수 있을 것이다.

1차 주행의 알고리즘

Speedy-3의 1차주행 알고리즘은 Speedy-2의 것과 근본적으로는 같고, 대략 다음과 같은 것이다.

1) 현재 있는 곳의 벽 정보를 얻어서 갈 방향을 찾는다. 이때 방향은 다음의 세가지 조건을 만족해야 한다.

a) 그 방향에 벽이 없어야 한다.
b) 진행할 구간이 가본적이 없는 곳이라야 한다.
c) 진행할 구간이 루프 테스트에 의해 루프를 이루지 않아야 한다.

2) 만일 갈 방향이 하나이면 그곳으로 이동한다. 둘 이상이면 한 방향을 골라서 이동하고 나머지 갈 수 있었던 구간들을 목적지 리스트에 추가한다. 만일 이동한 구간이 이미 그 전에 목적지 리스트에 들어 있었으면 이때 리스트에서 삭제한다.

3) 만일 갈 방향이 하나도 없다면 최단거리 서브루틴에 의해 목적지 리스트에 들어 있는 곳 중에서 가장 빨리 갈 수 있는 곳을 찾아서 그곳으로 간다. 그전에 그동안 추가로 알아낸 정보에 의해 루프를 이루는 목적지가 있는지 검사하고 그런 곳이 있으면 목적이 리스트에서 삭제한다.

4) 만일 이동한 구간이 원래 대회에 규정되어 있는 목적지, 즉 가운데의 4구간중 하나이면 일단 시작점에 돌아갔다 온다.

5) 모든 가봐야 할 곳을 다 가보았으면 (즉 목적지 리스트가 비었으면) 시작점으로 다시 돌아가서 3차 주행을 하고, 시작점으로 돌아와 정지한다.

위의 알고리즘을 구현하려면 상당한 분량의 프로그램이 필요하다.

또 위의 많은 판단을 가능하면 정지하지 않은 채로 하는 것이 바람직하기 때문에 프로그램의 실행속도도 빨라야 한다.

Speedy-3의 이부분 프로그램은 Speedy-1에서 그 근원을 찾을 수 있고, 무수히 디버깅된 형태이기 때문에 흔히 미국사람들이 말하는 스파케티 코드, 즉 이리 저리 Go to가 있는 복잡한 것이 되어 버렸다. 하옇튼 전체적인 알고리즘은 위와 같으므로 제작에 있어 참조하기 바란다.

기본적인 주행

주행을 하면서 항상 고려해야 할 일은 다음과 같다.

첫째, 스텝모터의 다음 펄스 시간폭을 계산해서 타이머에 로드해야 한다. 최고 속도를 대략 1500pps라고 보면 두 모터에 대해 시간당 최대 3000번의 계산이 있어야 한다. 이 계산은 출발지점으로부터의 거리와 목적지점까지의 거리, 방향오차의 유무에 따라 이루어진다.

둘째, 각 센서의 처리를 해야 한다. 고속 주행시에도 적절한 정도의 정밀성을 갖기 위해 센서의 샘플링은 일정 속도 이상이어야 한다. 적외선 센서의 경우 각 센서에 대해 초당 1000번을 샘플링하고, 4조의 센서를 시분할로 사용하므로 CPU쪽에서 보면 초당 4000번 처리를 해야 한다.

이 두가지가 3개의 타이머에 의해 인터럽트 처리되므로 CPU는 최고 초당 7000번 까지의 인터럽트 서비스를 해야 하고, 다른 일을 전혀 안하더라도 한 인터럽트에 대해 약143μs밖에 여유가 없다. 8MHz로 동작하는 V40 CPU의 한 명령당 시간이 평균 2μs정도라고 가정하면, 인터럽트 자체에 의한 시간지연이 없다고 해도 각 인터럽트가 약 70 명령어 밖에 수행 못한다는 결과가 된다. 실험해본 분석이지만 어쨌든 이 정도의 속도도 CPU에는 상당한 부담이 되고 따라서 Turbo-4의 경우처럼 센서의 처리는 다른 CPU가 맡도록 하는 것도 좋을 것이다. DMA(Direct Memory Access)등을 사용하면 좋겠지만, 하드웨어가 복잡해지므로 역시 문제가 있다. 그래서 Speedy-3에서는 인터럽트에 의한 처리를 최대한 간단히 하고, 다른 처리는 주 프로그램에서 했다. 메인 프로그램은 대부분의 시간동안 같은 부분을 계속 반복 수행하는데 일단 메모리에 저장된 센서의 상태를 분석해서 방향수정이 필요한지를 본다. 만약 필요하면 속도를 늦춰야 하는 쪽의 플래그를 세트한다.

가 · 감속 및 오차보정

스텝모터의 인터럽트 서비스 루틴은 매번 거리 카운터를 증가시키면서 그에 따른 펄스폭의 상수를 테이블에서 가져 오는데, 이때 방향오차를 나타내는 플래그가 세트되어 있으면 저속일때 1 / 8, 고속일때 1 / 16 만큼 속도상수를 크게 해서 그쪽의 속도를 낮춘다. 이는 계산을 쉽게 하기 위해 대략 정한 값이다. 이 값이 커지면 보정을 빨리 할 수 있으나 탈조의 위험이 커지고 속도를 더 여유있게 낮춰야 한다. 어느 이상 보정을 많이 하면 오히려 불안정해진다. 이는 자동제어 시스템에서 흔히 볼 수 있는 현상인데, 시간이 있으면 보다 엄밀한 분석을 해서 어떤 보정방법이 좋은지를 이론적으로 구할 수도 있다. 주행중 해야 하는 오차보정으로는 방향보정과 거리보정이 있다. 긴 거리를 방향보정을 해가면서 가다보면 아무래도 똑바로 가지 못한 만큼 거리의 오차가 생긴다. 이를 보정하기위해 옆벽이 끝나는 시점에서 거리보정을 한다.

위에서 언급한 모든 일은 직진 주행시 일어나는 일들이다. 그리고, 곡선회전이 아닌 제자리에서 하는 일반적인 회전의 경우에는 회전하는 동안에 아무런 보정도 하지 않는다. 다만 회전이 일단 끝난 후에 센서의 상태를 보아 정확히 회전하지 않은 것이 발견되면 그만큼 다시 회전을 한다.

1차 주행시 다음 구간으로 들어서면서 어떤 시점에서 그 구간의 벽 정보를 판별해서 그 다음에는 어느 쪽으로 갈 것인지를 결정해야 한다. 만일 계속 직진하게 되면 일정한 속도를 유지하면서 그대로 간다. 여기에서 거리로 보면 잠시 더 가속을 할 수도 있지만 그렇게 하지 않는데 그 이유는 실제로 그렇게 하면 계속 가감속이 반복되어 주행이 매우 불안정해지기 때문이다. 따라서 1차 주행에서는 이미 아는 길을 가는 경우를 제외하고는 최고속도를 제한한다. 직진하지 않고 회전해야 할 경우에는 그 구간의 중앙까지 진행한 후에 정지해서 회전한다. 만일 가볼 방향이 없다면 일단 감속해서 정지한 후에 최단거리 서브루틴에 의해 안가본 곳 중 가까운 곳을 찾아간다. 벽 정보의 판별은 매우 중요한 문제이고 한번 잘못되면 돌이킬 수 없는 것이기 때문에 여러번 데이터를 비교해서 많은 쪽으로 결정한다. 이에 대한 자세한 것은 프로그램 리스트를 참조하기 바란다.

최단거리경로 탐색 프로그램

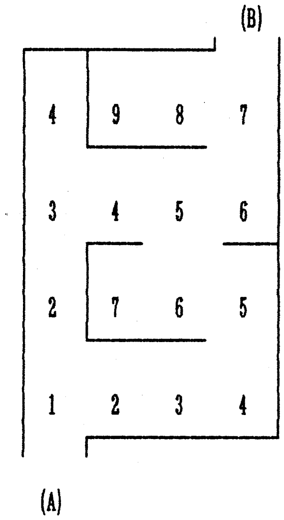
Speedy-3에서는 1, 2차 주행을 막론하고 이미 알고 있는 두 지점간의 이동은 모두 최단거리경로 서브루틴에 의하고 있다. 여기서는 등고선법이라고 흔히 불리우는 방법을 사용한다.

그림 1을 보면 A지점에서 B지점까지의 지름길을 찾기 위해 A지점에서부터 거리에 따라 차례로 숫자를 써간다. 갈림길이 나오면 여러 갈래로 동시에 숫자를 매긴다. 막힌 길이 나오거나 이미 숫자가 매겨져 있으면 그곳에서 멈춘다. 그림 1에서는 8스텝에서 목적지에 도착함을 알 수 있다.

여기에서 지름길을 찾기위해서 반대로 B지점에서 출발해서 숫자가 작아지는 쪽으로 찾아가면 A지점까지의 지름길을 찾을 수 있다. 이 경로를 저장했다가 다시 반대로 주행하면 된다.

한편 최단거리가 아니고 최단시간경로를 구하기 위해 많은 참가자들이 노력했으나 좋은 결과는 없었다. 정말로 최단시간경로를 구하기 위해서는 모든 경로의 시간을 계산해 볼 수 밖에 없는데 이는 최적화 문제로 유명한 세일즈맨 문제와도 유사한 것으로서 그 계산량이 방대하기 때문에 구현하기가 어렵다. 그보다는 등고선법의 최단거

〔그림 1〕 등고선법



리경로 알고리즘을 약간 변경해서 부분적으로 성공한 경우가 있었는데 Turbo- 4도 이러한 경우이다.

필자도 대회가 끝난 후에 Turbo-4와 비슷한 방법으로 회전할 때마다 한 스텝씩의 지연을 두어 결과적으로 한 구간 회전이 두구간 직진에 걸리는 시간과 같아지도록 등고선법을 고쳐서 여러 미로에 대해 시뮬레이션을 해보았다. 제6회 대회의 미로에 대해서는(그전 대회에서는 최단거리경로와 최단시간경로가 같았다) Trubo-4와 동일한 결과를 얻었고 다시 Back Trace 부분을 약간 개량해서 최단시간경로를 얻을 수 있었다. 물론 이는 특정 미로에 대한 결과이고, 다른 미로에서도 최단시간경로를 제대로 얻을 수 있다는 것은 아니다. 그러나 대부분의 경우에 있어 적어도 최단거리 방법보다는 좋은 결과를 얻을 것으로 보였다. 이 방법은 등고선법을 약간만 개량하면 된다. 또한, 최단시간경로의 계산시간도 등고선법과 거의 비슷하므로 앞으로 많은 적용이 있을 것으로 기대된다.

루프 테스트

루프 테스트란 어떤 구역이입구가 하나밖에 없고, 그 내부에 목표지점이 존재하지 않는다면 당연히 그 안으로는 들어가 볼 필요가 없다는 개념에 의한 것이다. 개념자체는 간단한 것이고, 그 구현도 그다지 어렵지는 않지만 그동안 사용된 예가 거의 없는 이유는 1차 주행시간이 기록에 포함되지 않아서였을 것이다. 또 이 테스트는 1차 주행이 이루어지는 동안 다음 갈 곳을 판별하는 시점에서 매우 빠르게 계산이 이루어져야 하기 때문에 그 적용을 망설인 경우도 있었을 것이다. 그러나 알고리즘 자체는 비교적 간단한 것이므로 잘 활용하면 사용상 지장은 없다.

실제 적용에 있어서 루프 테스트 자체는 미로의 크기가 어느정도 이상이 되어야 적용 가능한 것이고, 또 목적지가 루프에 포함되는지의 테스트도 작은 미로에서는 어렵기 때문에 개인적으로 대회에 참가하는 사람은 디버깅하는데 많은 어려움이 있다. 일단 시뮬레이션에 의해 확실히 디버깅한 후에 그것을 가능한한 그대로 로보트 프로그램에 옮겨 넣어야 할 것이다. 루프 테스트의 구체적인 방법을 알기위해 아래의 프로그램을 예로 들었다(리스트 1 참조).

아래 프로그램은 파스칼로 쓰여진 마이크로 로보트 시뮬레이션 프로그램의 일부분인데 cx, cy지점에서 d방향

〔리스트 1〕

```
Function loop( cx, cy : wallNtype ; d : dtype ):boolean;
  var
    gd, td, tdl : dtype ; { test direction }
    x, y : wallNtype ;
    q : byte ;
    c : integer ;
begin
  advance( cx, cy, d, x, y ); { x, y is advanced position of cx, cy to d
                                direction }
  td:=tdir( right, d );       { turn direction d to right }
  loop:=false;
  if sectorknown[x,y] then exit;
  q:=0;
  c:=0;
  repeat
    case wall[x,y,td] of
      exist : begin
              td:=tdir( left, td );
              inc(c);
              end;
      unknown, nonexist :
              begin
              advance( x, y, td, x, y );
              td:=tdir( right, td );
              dec(c);
              q:=q or 1 shl (ord(x<8) shl 1 or ord(y<8));
              end;
    end;
  until SectorKnown[x,y];
  loop:= (x=cx) and (y=cy) and (q<>15) and (c>0) ;
end;
```

으로 루프가 형성되는지를 보는 함수이다. 관심을 갖고 조금만 연구하면 원리를 이해할 수 있을 것이다. 위에서 q라는 변수는 루프를 테스트하면서 미로의 4개의 사분면을 모두 지났는지를 검사하기 위한 것이고, 이에 의해 루프에 골이 포함되어있는 지를 알 수 있다. c는 탐색의·회전방향을 확인하기 위한 것이다. 프로그램을 보면 알 수 있겠지만 루프테스트를 적용하기 위해서는 벽의 유/무 뿐만 아니라 그 벽의 존재를 인식하는 여부도 기록해 놓아야 한다. 또, 주행을 시작하기 전에 미리 미로의 가장자리 벽 정보를 입력해 놓아야 한다. 그렇지 않으면 테스트가 엉망이 되어버린다. 확실히 이해하기 위해서는 위의 프로그램을 따라가면서 직접 실행해보기 바란다.

프로그램의 개발

Speedy-3의 모든 개발은 IBM-PC 상에서 이루어졌다. 먼저, 미로탐색 알고리즘의 시뮬레이션 프로그램이 터보파스칼에 의해 작성되어 여러 알고리즘에 대한 테스트를 했다. Speedy-3상에 실릴 프로그램은 마이크로소프트사의 매크로어셈블러와 링커에 의해 .EXE 화일의 형태로 만들어진 후 EXE2BIN 프로그램에 의해 .BIN 화일로 변환되어 RS-232C에 의해 전송되었다. 의심스러운 변수의 값은 수시로 PC로 전송하도록 해서 그 값을 확인했다. 속도상수 테이블은 Lotus 1-2-3에 의해 만들었고, 실험에 의해 적절한 가속도를 찾았다. 초기 가속도가 너무 크면 미끄러질 우려가 있고 후기 가속도는 토크의 부족으로 크게 할 수 없기 때문에 주로 중간단계에서 가속을 많이 했다. 회전의 경우에도 같은 테이블을 사용했는데 대부분 적절한 것 같았다.

뒤에 첨부한 프로그램 리스트 2는 대회때 Speedy-3에 실렸던 그대로이다. 리스트에 주석을 달았지만 디버깅을 한 후 정리되지 않았고 원래 어셈블리 프로그램은 얼핏 보고 이해하기 쉬운 것은 아니므로 일일이 자세한 부분을 이해하거나 그대로 옮겨 사용하려 하지 않기 바란다. 변수나 서브루틴 등에 중복된 기능이나 사용 안된 부분이 있는 것을 양해하기 바란다.

맺음말

여러해 째 마이크로 로보트 대회에 참가했으나 지겹다기 보다는 그때마다 다음에는 더 잘할 수 있을텐데 하는 아쉬움이 남는다. 이 대회에 참가함으로서 실제의 시스템을 개발하는 경험을 얻을 수 있다는 잇점이 있으나, 그보다도 대회 자체의 매력 때문에 지금껏 많은 시간과 노력을 투자하게 되었다. 필자는 이번 기회에 책에 나와있는 지식보다도 대회에 참가해서 얻은 경험을 가능한한 많이 전달하기 위해 노력했으나 역시 미흡한 점이 많은 것 같다. 앞으로 대회에 참가하고자 하는 사람들에게 한가지 하고싶은 말은 가능한한 시간여유를 많이 두고 여러가지로 실험을 해보라는 것이다. 노력하는 만큼의 성과가 있을 것이다.

지면 관계상 전체 프로그램을 다 싣지못하는 점을 양해바라며 프로그램이 필요한 분은 본사 편집부 제작코너 담당자 앞으로 연락바란다. PC

〔리스트 2〕 최단거리경로계산 프로그램

```
        ; FSP.ASM
        ; Find Shortest Path
        ; 등고선법에 의한 최단 경로의 계산.

        include vars.inc
        include params.inc
        include defs.inc

cseg    segment public 'CSEG'
        assume cs:cseg, ds:dseg

        public fsp, trace

        extrn convtd:near, convdt:near, advnce:near, error:near
        extrn turnleft:near, turnright:near, turnend:near, gotoend:near
        extrn stmotor:near, travel:near, turnback:near, spedtbl:near

fsp     proc near
        ; find shortest path subroutine
        ; 여기서는 가상 마우스의 개념을 도입해서 최단 경로를 찾는다.
        ; 알고리즘의 자세한 것은 본문을 참고할 것.
        ; initialize mouse list
        ; back trace buffer, steps matrix

        mov di, offset steps
        mov cx, 320
        xor ax, ax
        cld
        rep stosw
```

```
        ; check if the mouse is in one of
                  the destination list
        mov di, offset dlist
        mov al, cloc
        mov cx, 128
fsp01:  cmp al, [di]
        je fsp02
        inc di
        loop fsp01
        jmp fsp03
fsp02:  mov byte ptr [di], 0 ; erase the
                  item from dest. list.
fsp03:
f100:   ; initialize first mouse
        mov al, cloc
        mov bx, offset mlist
        mov [bx], al
        mov byte ptr 128[bx], 1
        mov bl, al
        xor bh, bh
        mov steps[bx], 1 ; current step
        mov cstp, 2
f10:    mov mno, 0
        mov tno, 80h
f11:    call mvemouse
        inc mno
        dec tno
        jnz f11
        inc cstp
        call fod
        jnc f10
        ; found destination
        mov bl, destloc
        mov bh, 0
        mov al, cstp
        dec al
        mov cstp, al
        mov steps[bx], al

        ; back trace
        mov di, offset tracb
        mov dir, 1
        mov al, destloc
        mov loc1, al
        dec cstp
        call getdir
        mov dl, dir
b1:     ; repeat until arrive
        mov bl, loc2
        mov bh, 0
        cmp steps[bx], 1
        jz b11
        dec cstp
        mov dh, dl
        call getdir
        mov dl, dir
        call convdt
        mov [di], al
        inc di
        jmp b1
        ; start position
b11:    mov al, 2
        call convtd
        mov dl, cdir
        call convdt
        mov byte ptr [di], al
        ret
fsp     endp

getdir  proc near
        ; back trace 때 갈 방향을 찾는다.
        push ax
        push bx
        push dx
        call chkdir
        jnz gd2 ; go straight
        ; check left
        mov dl, dir
        mov al, 1
        call convtd
        mov dir, dh
        call chkdir
        jnz gd2
        mov dl, dir
        mov al, 2
        call convtd
        mov dir, dh
        call chkdir
        jnz gd2
        mov dl, dir
        mov al, -1
        call convtd
        mov dir, dh
        call chkdir
        jnz gd2
        ; 물론, 프로그램이 정상적으로 동작하면 아래의
          명령이 실행될 리가 없다. 이는 개발 단계에서
          프로그램의 오류를 발견하기 위한 것이다.
        call error
        db 'GETDIR found no way to go.', 0
gd2:    mov al, loc2
        mov loc1, al
        pop dx
        pop bx
        pop ax
        ret
getdir  endp

chkdir  proc near
        ; getdir 에서 각 방향의 test 를 한다.
        mov al, dir
        mov bl, loc1
        mov bh, 0
        and al, wall[bx]
        jz cd2 ; wall exist
        mov al, dir
        mov dl, loc1
        call advnce
        mov loc2, dh
        mov bl, dh
        mov al, cstp
        cmp al, steps[bx]
        jz cd1     ; go to that direction
cd2:    xor al, al ; set zero flag
        ret
cd1:    xor al, al
        inc al  ; reset zero flag
        ret
chkdir  endp

fod     proc near
        ; found destination
        ; 여러 가상 마우스중 하나라도 목적지에 도달했
          는지 검사. 여기서 목적지도 복수이다. 실제로
          검사는 목적지 리스트에 들어있는 위치에 step
          수가 기록되었는지를 본다.
        push bx
        push dx
        push si
        mov si, offset dlist
        mov cx, 128
        mov bh, 0
fod3:   mov al, [si]
        or al, al
        jz fod1
        mov bl, al
        mov al, steps[bx]
        or al, al
        jnz fod2
fod1:   inc si
        loop fod3
        pop si
        pop dx
        pop bx
```

```
        clc
        ret
fod2:   ; found
        mov destloc, bl
        mov byte ptr[si], 0
        pop si
        pop dx
        pop bx
        stc
        ret
fod     endp

mvemouse        proc near
        ; 가상 마우스를 움직이는 서브루틴. 현재의 step 수를 기록하면서 이동
        ; 한다. 이동중 이동 가능한 방향이 여럿이면 추가로 새로운 가상 마우스
        ; 를 만들어 그곳으로 가게 한다.
        ; 가상 마우스가 '죽는' 경우는 막힌 길로 들어섰을 때와 이미 다른
        ; 가상 마우스가 지나가서 step 을 기록해 놓은 경우이다. 모든 경로에
        ; 대해 주행 시간을 계산하는 경우에 비해서 뒤쪽 조건 때문에 계산시간이
        ; 훨씬 짧아지게 된다. 그러나 근본적으로 이 방법으로는 거리가 짧은
        ; 경로를 찾아낸다는 문제점이 있다. 본문에서 설명되지만, 한번 turn 때
        ; 마다 한 step 씩의 delay 를 주면 근사적으로 최단 시간 경로를 찾을
        ; 수 있는데 아래의 프로그램에서는 구현되어있지 않다.
        mov bl, mno
        xor bh, bh
        mov ah, mlist[bx]
        test [bx+mlist+128], 1
        jnz mv0
        ret     ; dead mouse
mv0:    mov bl, ah
        mov al, cstp
        cmp al, steps[bx]
        jnz mv1
        ret
mv1:    mov fone, 0
        mov loc1, ah
        mov dir, 1000b
mml1:   mov bl, loc1
        mov al, wall[bx]
        and al, dir
        jz mme1
        mov al, dir
        mov dl, loc1
        call advnce
        mov loc2, dh
        mov bl, dh
        or bl, bl
        jz mme1
        cmp steps[bx], 0
        jnz mme1
        test fone, 1
        jz mmj1
        call newmouse
        jmp mme1
mmj1:   mov fone, 1
        mov bl, mno
        mov bh, 0
        mov al, loc2
        mov mlist[bx], al
        mov bl, al
        mov ah, cstp
        mov steps[bx], ah
mme1:   shr dir, 1
        jnz mml1
        cmp fone, 0
        jz mv2
        ret
mv2:    mov bl, mno
        mov [bx+mlist+128], 0
        ret
mvemouse        endp

newmouse        proc near
        push ax
        push bx
        push cx
        cmp loc2, 0
        jz enm
        mov bx, offset mlist + 128
        mov cx, 128
        xor al, al
nm1:    cmp al, [bx]
        jz nm2
        inc bx
        loop nm1
nm2:    mov byte ptr [bx], 1
        sub bx, 128
        mov al, loc2
        mov [bx], al
        mov bl, al
        xor bh, bh
        mov al, cstp
        mov steps[bx], al
enm:    pop cx
        pop bx
        pop ax
        ret
newmouse        endp

trace   proc near
        ; fsp 에 의해 갈 목표지점이 찾아지고 그 곳까지의 trace buffer
        ; 가 만들어졌으면 이제 그곳으로 간다. 곡선 회전등을 위해서는
        ; 이 서브루틴을 새로 만들면 된다.
        ; trace buffer 에는 절대방향 ( 동/서/남/북 ) 이 기록되어 있기
```

```
        ; 때문에 상대 방향 ( 좌/우/직진 ) 으로의 변환이 필요하고
        ; 같은 방향이 계속될 경우 이를 하나로 합쳐서 고속 주행을 하도록
        ; 한다. 이 서브루틴은 새로 도착한 구간의 벽정보를 읽을 수 있는
        ; 시점에서 return 한다.
        mov fast, 1
tr1:    cmp di, offset tracb-1
        jnz t0
        mov fast, 0
        ret
t0:     mov al, [di]
        mov dl, cdir
        call convtd
        mov cdir, dh
        mov dir, dh
        mov al, dh
        mov dl, cloc
        call advnce
        mov cloc, dh
        cmp byte ptr[di], 0
        jz t2
        cmp byte ptr[di], -1
        jz t1
        cmp byte ptr[di], 1
        jz tleft
        ; turn back
        call turnback
        call turnend
        jmp t2
t3:     dec di
        jmp tr1

t1:     ; turn right
        call turnright
        jmp tl11
tleft:  call turnleft
tl11:   call turnend
t2:     ; go straight
        mov al, 00001000b
        out portA, al
        mov cx, sectdist
t6:     cmp di, offset tracb
        jz t5
        cmp byte ptr[di-1], 0
        jnz t5
        dec di
        add cx, sectdist
        mov al, cdir
        mov dl, cloc
        call advnce
        mov cloc, dh
        jmp t6
t5:     mov r_dist, cx
        mov l_dist, cx
        xor ax, ax
        mov r_count, ax
        mov l_count, ax
        mov fast, 1
        call stmotor
        cmp di, offset tracb
        jnz t52
t51:    call travel
        mov ax, r_dist
        sub ax, r_count
        cmp ax, sectdist-sensepos
        jae t51
        jp t3

t52:    call gotoend
        jp t3
trace   endp

cseg    ends
        end
```

PC-매거진

박희정/명지대 국문과

퍼블릭 도메인(Public Domain)이라고 하는 것은 독자들도 익히 알고 있듯이 상업적인 목적을 떠나서 모든 사람들이 사용하도록 하는 공개 소프트웨어를 말한다.

이러한 공개 소프트웨어는 유럽이나 미국 등지에서 성행하고 있는 것이며, 우리나라에서도 점차로 늘어나는 추세에 있다. PC 통신에 있어서 파발마, 리볼트, 메아리 등은 그 좋은 예라 하겠다. 앞으로도 모든 방면에서 좋은 공개 소프트웨어가 나올 것으로 기대된다.

여기에 소개하는 PC-매거진(PC-Magazine)도 공개 소프트웨어의 한 가지로 간략히 소개한다면 그 이름에서도 보는 바와 같이 PC를 사용하는데 있어서 필요한(또는 유용한) 유틸리티들을 모아 놓은 것이다.

PC-매거진은 처음에는 디스켓 1장에 PCUTL.EXE와 MANUAL.EXE라는 두개의 화일만이 존재하지만 두 프로그램을 실행시키면 총 41개의 유틸리티가 만들어진다. 이것은 일반적으로 통신에서 많이 사용하는 방법으로 프로그램이 길어지면 전송하는 시간 또한 길어지므로 어떤 논리에 의하여 화일을 축소(Archive)시켜 전달하고 이것을 다시 확장하는 기술을 이용하고 있다. 이 기법은 통신분야뿐만 아니라 여러 분야에서도 많이 쓰이고 있다.

그럼 PC-매거진에는 어떤 유틸리티가 담겨 있는지 알아보기로 하자. 표 1은 PC-매거진을 모두 전개한 디렉토리와 각 유틸리티들의 역할을 요약하

〔표 1〕 PC매거진 디렉토리

PCUTIL	.EXE	PC매거진을 전개하는 프로그램
MANUAL	.EXE	매뉴얼을 전개하는 프로그램
ATTR	.COM	화일의 속성을 보여주거나 변경한다.
SETUP	.COM	프린터의 제어코드를 언제든지 변환할 수 있는 램상주 유틸리티이다.
SIZE	.COM	화일의 용량을 검사한다.
STATLINE	.COM	화면의 26라인에 Shift 키의 상태를 보여준다.
UNDEL	.COM	지워진 화일을 복구하여 준다.
UNLOCK	.COM	LOCK.COM으로 걸려진 패스워드를 풀어주는 프로그램
VTREE	.COM	서브 디렉토리의 경로를 도면화하여 보여준다.
WAITASEC	.COM	화면의 지나간 부분을 다시 내려 볼 수 있는 램상주 유틸리티
WELCOME		HELF 메세지
WHERE	.COM	찾고자하는 화일의 경로를 모두 보여준다.
XDEL	.COM	화일을 선택적으로 지워주는 프로그램
MANUAL		PC매거진의 매뉴얼
BAC	.COM	하드디스크 또는 플로피디스크의 한 디렉토리 안에 있는 모든 화일을 백업한다.
BOOTSCRN	.	Boot Screen
BROWSE	.COM	화일의 내용을 화면에 보여준다.
COLOESET	.COM	컬러 색상을 조정한다.
DDIR	.COM	알파벳 순으로 정렬된 화면을 두 줄로 화면에 출력하여 준다.
DISKSCAN	.COM	디스크의 이상유무를 살펴본다.
DOORS	.COM	지워진 화면을 다시 보여주는 램상주 유틸리티
DOS-EDIT	.COM	커서를 마음대로 움직여 도스 명령어를 다시 입력하지 않고 편집할 수 있게 해준다.
FREE	.COM	디스크의 용량을 알아보는 프로그램
KEEPER	.COM	최근에 지나간 10개의 명령어를 다시 사용할 수 있다.
KEY-FAKE	.COM	프로그램 안에서 키를 두드리지 않고 매뉴 선택 등을 가능하게 해준다.
LOCATE	.COM	디스크안에 찾고자 하는 정보가 어느 곳에 있는지를 살펴본다.
LOCK	.COM	화일에 암호를 만들어 다른 사람이 사용하지 못하게 하여 준다.
LPTPORT	.COM	프린트 포트를 변환하여 주는 유틸리티
MONOGRAF	.DRF	LOTUS'S 1-2-3(Version 1A)상에서 모노크롬모니터를 지원하는 그래픽 드라이버
MOVE	.COM	화일들을 선택적으로 복사한다.
NO	.COM	화일의 삭제나 복사의 경우 특정 화일을 제외시킨다.
NPAD	.COM	화면 우측 상단에 작은 윈도우를 만들어 준다.
POPDIR	.COM	PUSHDIR.COM으로 지정된 디렉토리로 돌아간다.
PR	.COM	화일을 프린터로 출력할 경우 그 포맷을 잡아준다.
PRSWAP	.COM	그래픽 프린터가 아닌 프린터를 사용할 경우 IBM-텍스트 그래픽을 변환하여 준다.
PUSHDIR	.COM	현재 있는 경로를 저장한다.
QUICKEYS	.COM	키보드 입력을 좀더 빠르게 조절하여 준다.
RENDIR	.COM	서브 디렉토리의 이름을 사용자가 정의하는 대로 변경한다.

여 나타낸 것이다.

PC-매거진을 전개하려면 빈 디스켓이나 하드디스크의 서브디렉토리 안에 두개의 화일(PCUTL.EXE와 MANUAL. EXE)만 복사해 넣는다. 그런다음 다음과 같이 실행시키기만 하면 된다.

```
C:\TOOL>PCUTIL
```

또는

```
A:\>PCUTIL
```

매뉴얼을 참조하고 싶으면 다음 명령을 실행한다.

```
A:\>BROWSE MANUAL
```

또는

```
A:\>COPY MANUAL PRN
```

로 프린터로 출력하여 참조할 수 있다. 필자의 경우 매뉴얼을 프린팅한 결과 총 46페이지의 (정말 매뉴얼다운) 엄청난 분량이 나왔다. 이렇게 많은 분량을 모두 지면에 설명하기에는 불가능하고 PC를 사용하는 데 있어서 유용하게 쓰일 수 있는 유틸리티만 몇개 골라 소개하고자 한다.

BROWSE.COM

이것은 워드프로세서로 작성된 화일을 화면상에 출력하는 프로그램이다. 물론 DOS의 내부명령어인 TYPE이나 COPY 등의 명령으로도 가능하지만 스크롤된 부분을 다시 내릴 수는 없다. 그래서 볼랜드사를 위시한 여러 회사에서는 도큐먼트화일을 볼 수 있게 README. COM을 디스켓에 같이 실어주고 있다. 그러나 이것은 그 프로그램에만 한정되어 있고 또 폭이 80컬럼이어서 80컬럼 이상인 것은 라인피드가 되어 버리는 단점이 있다. 이에 반해 BROWSE.COM은 80컬럼의 제한된 모니터에서 그 이상의 문자를 볼 때에는 우측으로 스크롤할 수 있다는 장점을 지니고 있다. 형식은

```
BROWSE [d:][path] filename[.ext]
```

과 같으며 화면을 움직이기 위해 통상적으로 많이 쓰이는 4개의 커서키와 PgUp, PgDn이 사용되므로 사용법도 간단하다.

우측으로 스크롤할 경우는 8컬럼씩 움직이게 되어 있으며 DOS디스켓에 있는 DEBUG.COM을 이용하면 이 이동폭도 쉽게 변경할 수 있다. 다음과 같이 DEBUG.COM과 BROWSE.COM을 같이 로드한 후 E(Enter)명령으로 오프셋번지 10F의 내용을 임의의 수치(16진수)로 바꾸면 된다.

```
A:\>DEBUG BROWSE.COM
-E 10F
-xxxx:010F 08.28
-W
-Q
A:\>
```

위에서 28H은 10진수로 40이며 오른쪽으로 스크롤할 경우 40컬럼을 움직이게 된다. 즉 80컬럼을 움직이게 하려면 28H대신에 50H(10진수로 80)을 입력하면 된다.

KEY-FAKE.COM

KEY-FAKE.COM은 필자가 사용한 결과 정말 유용한 도구였다. 일반적으로 배치화일에 여러가지 기능을 추가하는 유틸리티(즉, 배치화일에서 메뉴를 선택할 수 있게 해 주는 방법이라든지 배치화일을 보다 보기좋게 작성하는 방법등)가 나왔지만 정작 프로그램 내에서 실행되는 유틸리티는 아직 보지를 못했다.

KEY-FAKE. COM은 이러한 것을 가능하게 해준다. 한 예로 보석글을 사용할 경우 우리는 항상 PATH라 하여 문서를 읽거나 보관될 경로를 설정하여 주어야 한다.

이런 경우 보석글 프로그램 앞에서 이 유틸리티를 실행하면 자동적으로 이러한 설정을 대신해 준다. 그러면 KEY-FAKE. COM의 형식과 기능을 알아보자.

형식은

```
KEY-FAKE ["xyz"] [nn] [0] [@F]
Programname
```

과 같다.

"xyz"는 일반 문자열을 입력받을 경우 사용된다. 문자는 하나 또는 여러개가 가능하며 반드시 ""안에 있어야 한다. [nn]은 ASCII의 제어코드 부분을 입력할 때 사용된다. 즉 리턴(CR : 13번)이나 탈출키(ESC : 27번) 등을 동작시킬 때 유용하다.

[O]는 키보드 버퍼를 체크해야 하는 경우 사용되며 키보드 버퍼가 비어있는지 아니면 키입력이 제대로 성공했는지를 판별한다.

[@F]는 ASCII 코드가 아닌 키의 입력에 사용된다. 이것은 ASCII가 아닌 예로 커서(Cursor)키를 작동한다거나 펑션키(Function Key)를 작동할 경우 사용된다.

위의 기능을 잘 활용한다면 편리한 유틸리티가 될 것이다.

만약 LOTUS 1-2-3를 사용한다면 다음과 같은 방법으로 배치화일을 만든다.

```
C:\>COPY CON RUNLOTUS.BAT
CD\LOTUS
KEY-FAKE 0 13 0 13 0 13 0 13 0 13 "/FR"
LOTUS
^Z

  1 file(s) copied

C:\>LOTUS
```

다른 예로 앞에서 말한 바와 같이 보석글에서 항상 경로를 정하는 번거로움을 없애는데도 사용된다.

```
C:\WORD\G2>COPY CON GEM.BAT
HGC FULL
NKP G
KEY-FAKE "PATH C:\WORD\G2\DATA" 13
```

```
G2
^Z

  1 file(s) copied

C:\WORD\G2>GEM
```

물론 KEY-FAKE.COM은 보석글과 같은 디렉토리 안에 존재하여야 한다. 아니면 앞부분에 경로를 지정하여도 좋다.

마지막으로 베이직(인터프리터 베이직)의 경우를 보면 처음 베이직을 기동시켰을 경우 바탕색과 글씨의 색을 마음대로 바꾸고 싶을 경우 이 유틸리티로 간단히 해결된다. 만약 파란색 배경에 노란글씨로 초기화되기를 원한다면 다음과 같이 배치화일을 사용하여 실행하면 된다.

```
C:\DOS>COPY CON BASIC-1.BAT
KEY-FAKE "COLOR 14,1,1" 13 "CLS" 13
BASICA
^Z

  1 file(s) copied

C:\DOS>BASIC-1
```

또 같은 방법으로 베이직의 펑션키도 사용자 마음대로 정의할 수 있다. 주의할 점은 배치화일의 이름이 화일내에 실행하는 화일이름과 같아서는 안된다는 것이다. 왜냐하면 화일이름이 같은 경우 확장자명이 COM, EXE, BAT 순서로 우선 순위를 갖고 있어서 .BAT보다 .COM이나.EXE가 우선 순위가 되므로 .BAT는 무의미하게 되어 버리기 때문이다.

한편 이 유틸리티의 키입력의 최대 범위는 124자이며 XyWrite Ⅱ와 같이 하드웨어 키보드 인터럽트를 사용하는 경우나 통신용 프로그램등에서는 사용이 불가능하다는 단점이 있기도 하다.

MOVE.COM

MOVE.COM은 도스의 내부 명령어인 COPY와 유사하나 복사하는 도중에 복사할 화일을 선택할 수 있는 특징을 지니고 있다.

COPY명령은 와일드카드(? 이나 *)를 사용하여 공통적인 부분을 가진 화일을 하나의 명령으로 처리하게 된다. 그러나 만약 그중에 불필요한 화일이 삽입되어 있다면 여간 골치거리가 아니다. 하나씩 복사하기도 불편하고 불필요한 것까지 복사하는 방법도 번거롭다. 물론 PCTOOLS라는 편리한 프로그램이 있기는 하지만 이 프로그램은 DOS프롬프 상태에서도 손쉽게 이런 문제를 해결해 준다.

MOVE.COM은 COPY명령과 동일한 방법으로 실행하나 복사하는 화일 하나마다 그것을 복사할 것인지를 물어보게 되어 있다.

입력형식은

```
MOVE [d:]filename[.ext] d:
```

과 같으며, 물론 화일이름에 와일드카드 사용이 가능하다.

앞에서 설명하였듯이 만약 A:\의 모든 베이직 프로그램 중 필요한 것만을 B:\로 복사한다고 하자. 그러면 다음과 같이 실행된다.

```
A:\>MOVE A:*.BAS B:
Copy TEST.BAS (Y/N)?Y
Copy SAMPLE.BAS (Y/N)?N
Copy TEST2.BAS (Y/N)?Y
:
```

이 유틸리티가 갖는 단점이라면 경로 지정에 있어서 경로 선택이 불가능하다는 점이다. 리스트 1은 필자가 MOVE.COM을 디스어셈블시켜 본 것인데 필자의 어셈블러 실력이 짧은지라 완벽한 소스는 만들지 못하였다. 남의 프로그램을 다시 크로스컴파일하여 원래의 소스를 만드는 것은 어렵거니와 다만 부분적으로 독자들에게 도움이 되었으면 한다. 관심있는 독자라면 완전한 소스를 만들어 보는 것도 바람직한 방법이라고 생각한다. 참고로 리스트 1은 DIS86.COM이라고 하는 disassembler 8086 / 88 Release 4.5를 이용하여 소스를 구성했음을 말해둔다.

NPAD.COM

우리가 보통 DOS를 사용하다 보면 무엇인가 암기해야 할 경우가 종종 생긴다. 보통은 필요한 사항들을 메모지에 기입하면서 참조하는데 NPAD.COM을 이용하면 간단히 해결할 수 있다. NPAD.COM을 이용하면 이런 문제가 쉽게 풀리게 된다. NPAD.COM은 화면의 우측 상단에 조그마한 윈도우를 만들어주는 램상주(TSR : Terminate and Stay Resident) 프로그램이다. 램상주 프로그램이란 프로그램 종료시 프로그램이 램안에 계속 상주하여 필요한 시기에 다시 호출되는 것을 말한다. 즉, 임의의 핫 키에 의해서 프로그램이 호출되는데 인터럽트벡터 테이블 중에서 인터럽트 9번을 이용하는 것이 보통이다. 실질적으로는 더 복잡하지만 간단히 이야기한다면 키가 눌러질 때마다 인터럽트 9번이 호출되는데 이것을 이용하여 키 입력을 체크하고 핫 키일 경우는 TSR 어플리케이션을 호출하게 되어 있다. 그렇지 않으면 정상적으로 리턴한다.

핫 키가 눌려지면 스크린 우측 상단에 10라인 25 캐릭터의 작은 윈도우가 형성된다(화면 1 참조). 작동하는 키는 일반적인 키와 역스페이스바 그리고 윈도우 전체를 지우는 Del키가 사용된다. 이들 램상주 프로그램들은 핫키(Hot-Key)에 의해서 임의로 호출되는데 여기서는 〈Ctrl〉-N이 사용된다. 만약 사용자가 이 핫 키를 바꾸고 싶다면 다음과 같이 DOS의 DEBUG.COM을 이용하여 간단히 변경할 수 있다.

```
A:\>DEBUG NPAD.COM
-E 336
xxxx:0336 0E.60
-E 337
xxxx:0337 31.29
```

-W
-Q

즉, 오프셋 어드레스의 xxxx : 0336은 핫 키의 ASCII 코드이고 xxxx : 0337은 스캔 코드가 저장되어 있는 곳이다. 위의 경우는 핫 키를 ASCII코드가 60H이고 스캔코드가 29H인 키로 전환이 된다. 또한 xxxx : 02FF와 xxxx : 0413의 데이터를 앞의 것과 같은 방법으로 수정하면 노트패드의 배경 색상과 글자의 색을 선택할 수 있다. 모노크롬 모니터를 사용한다면 상관이 없겠지만 컬러모니터를 사용한다면 노트패드가 흰색(7) 바탕에 검은(0) 글씨가 써진다. 이것은 앞의 두 메모리 값이 70h(십진수로 112)로 되어 있기 때문이다. 만약 흰 바탕에 빨간 글씨를 원한다면 70h를 74h로 수정하면 된다. 즉 앞의 4비트는 배경의 색상이고 뒤의 것은 글씨의 색상을 말한다. 표 2는 색상코드를 나타낸 것이다.

만약 사용자가 정의할 핫 키의 코드를 알아보려면 리스트 2의 베이직 프로그램을 실행시키면 쉽게 알아볼 수 있다.

PR.COM

〔리스트 1〕 디스어셈블 시킨 MOVE.ASM

```
CR    EQU     0DH
LF    EQU     0AH

CODE  SEGMENT PUBLIC
      ASSUME  CS:CODE,DS:CODE

      ORG     5CH
FCB1  DB      10H   DUP (?)   ;첫번째 FCB
FCB2  DB      14H   DUP (?)

      ORG     80H
DTA   DB      40H   DUP (?)   ;화일의 DTA부분

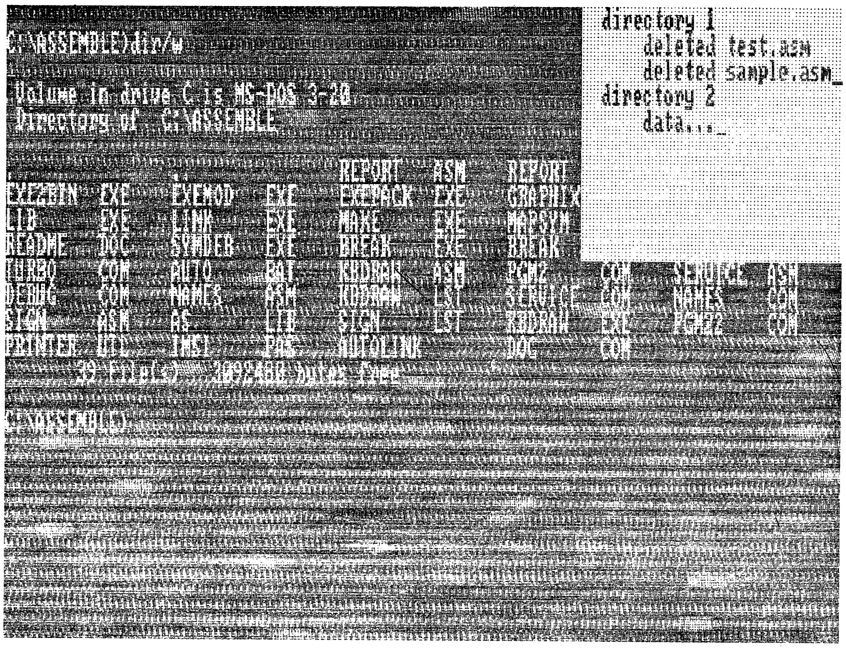
      ORG     0C0H
FCB3  DB      40H   DUP (?)

      ORG     100H      ;COM화일은 선두에 PSP가 존재하므로
      JMP     START     ;100H부터 시작
;
      LOGO    DB   '(C) S. HOLZNER 1984' ;저자의 LOGO
      DATA1   DB   37  DUP (0)
      DATA2   DW   ?
      DATA3   DW   ?
      DATA4   DW   ?
      DATA5   DW   ?
      DATA6   DW   ?
      MSG1    DB   CR,LF,'Copy $'
      MSG2    DB   ' (Y/N)?$'
      MSG3    DB   CR,LF,'Disk Full$'   ;Disk Full 에러
;
START:
      MOV   CX,002CH
      MOV   SI,OFFSET FCB2    FCB2에 설정된 화일이름을
      LEA   DI,[DATA1]        DATA1에 복사한다.
      REP   MOVSB
      MOV   DX,OFFSET FCB1
      MOV   AH,11H            ;최초의 디렉토리 엔트리를
      INT   21H               ;검사한다.
      CMP   AL,0FFH           ;디렉토리 엔트리가 존재한다면
      JNZ   X017B             ;X017B로 분기
      JMP   X02AD             ;그렇지 않으면 X024D로 분기
;
X017B:
      MOV   AH,9              ;MSG1을 출력
      LEA   DX,[MSG1]
      INT   21H               ;DOS기능 호출
      MOV   CX,000BH          ;화일이름을 출력
      MOV   BX,OFFSET DTA[1]
      MOV   AH,02H
X018B:
      MOV   DL,[BX]
      INC   BX
      INT   21H
      LOOP  X018B
      MOV   AH,09H        ;MSG2를 출력
      LEA   DX,[MSG2]
      INT   21H
      MOV   AH,01H        ;키보드부터 한 문자 입력
      INT   21H
      CMP   AL,59H        ;입력문자가 'Y'이면 분기
      JZ    X01A9
      CMP   AL,79H        ;입력문자가 'y'이면 분기
      JZ    X01A9
      JMP   X0298         ;다음 엔트리를 검색
;
X01A9:                    ;화일을 복사하는 부분
      MOV   CX,0025H
      LEA   SI,[DATA1]
      MOV   DI,OFFSET FCB3
      CMP   BYTE  PTR [SI+1],20H
      JNZ   X01C9
      PUSH  DI            ;레지스터를 스택에 보관한다.
      PUSH  CX
      LEA   DI,[DATA1]
      INC   DI
      MOV   CX,000BH
```

```
        MOV   AL,3FH
        REP   STOSB
        POP   CX              ;레지스터의 값을 다시 환원
        POP   DI
X01C9:
        MOV   BX,0000
        CMP   BYTE  PTR  [SI],3FH
        JNZ   X01DA
        MOV   BX,OFFSET DTA
        SUB   BX,OFFSET DATA1
        ADD   SI,BX
X01DA:
        MOVSB
        SUB   SI,BX
        LOOP  X01C9
        MOV   DX,OFFSET DTA   ;화일 오픈
        MOV   AH,0FH
        INT   21H
        MOV   DX,OFFSET FCB3  ;기능호출 16H에 의해 기록할 화일을
        MOV   AH,16H          ;새로 작성한다.
        INT   21H
        AND   WORD PTR [DATA2],0000H
        MOV   BX,008EH
        MOV   WORD PTR [BX],8000H
        MOV   BX,0090H
        MOV   AX,[BX]         ;[BX]의 값을 [DATA3]에 저장
        MOV   [DATA3], AX
        ADD   BX,02H
        MOV   DX,[BX]         ;[BX]의 값을 [DATA4]에 저장
        MOV   [DATA4],DX
        MOV   CX,0400H
        DIV   CX              ;몫은 AX에,나머지는 DX에 저장
        MOV   [DATA5],AX      ;몫을 [DATA5]에 저장
        TEST  DX,0FFFFH       ;각 비트의 AND를 취하고 결과에 따라
        JZ    X021D           ;    플래그를 세트
        INC   WORD PTR [DATA5]
X021D:
        MOV   BX,00CEH
        MOV   WORD PTR [BX],0400H
X0224:
        LEA   DX,[END-PRO]
        MOV   AH,1AH
        INT   21H
        MOV   DX,OFFSET DTA   ;첫째 화일을 레코드 단위로 읽어낸다.
        MOV   AH,14H
        INT   21H
        CMP   AL,00H
        JNG   X023D           ;정상적으로 끝나지 않으면 X023D로 분기
        OR    WORD PTR [DATA2],0001H
X023D:
        MOV   CX,0020H
        LEA   DX,[END-PRO]
        MOV   [DATA6],DX
X0248:
        MOV   DX, OFFSET FCB3  ;1레코드 단위의 데이터를 두번째 화일로
        MOV   AH,15H           ;    써넣는다.
        INT   21H
        CMP   AL,0
        JZ    X025E            ;에러 체크
        LEA   DX,[MSG3]        ;정상이면 X025E로 분기
        MOV   AH,09H           ;MSG3를 화면에 출력
        INT   21H
        JMP   X02AD

        NOP
X025E:
        DEC   WORD  PTR  [DATA5]
        JZ    X027C
        ADD   WORD PTR [DATA6],0400H
        MOV   DX,[DATA6]       ;새로운 DTA번지를 시스템에 설정
        MOV   AH,1AH
        INT   21H
        LOOP  X0248
        TEST  WORD PTR [DATA2],0001H
        JZ    X0224
X027C:
        MOV   AX,OFFSET DATA3
        MOV   BX,00D0H
        MOV   [BX],AX
        ADD   BX,02H
        MOV   AX,[DATA4]
        MOV   [BX],AX
        MOV   AH,10H           ;두번째 화일을 닫는다.
        MOV   DX,OFFSET FCB3
        INT   21H
        MOV   DX, OFFSET DTA   ;첫번째 화일을 닫는다.
        INT   21H
X0298:
        MOV   DX, OFFSET DTA
        MOV   AH,1AH           ;새로운 DTA번지를 설정
        INT   21H
        MOV   AH,12H
        MOV   DX,005CH         ;다음 디렉토리 엔트리를 검사
        INT   21H
        CMP   AL,0
        JNZ   X02AD            ;엔트리가 없으면 X02AD로 분기
        JMP   X017B
;
X02AD:
        RET
END-PRO:
CODE  ENDS
        END
```

〔화면 1〕 NPAD의 실행화면



〔리스트 2〕 핫 키의 코드를 찾는 프로그램

```
10 DEF SEG=&H40
20 A$=INKEY$:IF A$="" THEN 20
30 ADD=PEEK(&H1C):IF ADD=&H1E THEN ADD=&H3C
40 PRINT "ASCII CODE is ";HEX$(PEEK(ADD-2))
50 PRINT "SCAN  CODE is ";HEX$(PEEK(ADD-1))
60 PRINT
70 IF A$<>CHR$(27) THEN 20
80 END
```

워드프로세싱된 화일을 프린트할 경우 우리는 워드프로세서를 이용하는 것이 보통이다. 또 다른 방법으로는 간단한 도큐먼트를 출력하고자 할 때 도스 상태에서

```
A:\>TYPE filename.ext > PRN
```

혹은

```
A:\>COPY filename.ext PRN
```

등의 방법을 사용하였다. 혹자는 DOS에서 〈Ctrl〉-P를 누른 다음 화면상에 디스플레이 하여 그대로 프린터로 출력하기도 한다. 하지만 이 모든 방법은 각각 문제점이 있다. 워드프로세서(예를 들면 보석글 등)에서는 페이지 스킵이 자동적으로 일어나 연속용지를 사용할 경우 절단선 사이에 문자가 찍히는 것을 방지하고 있다. 그러나 위의 방법은 그러한 것을 무시한 채 무조건적으로 출력하기 때문에 여간 보기 싫은 것이 아니다. 또 보관하는데도 문제가 된다.

PR.COM은 도스상태에서 그러한 걱정을 없애주는 유틸리티이다. 프린트시 80컬럼 연속용지에 앞부분과 뒷부분으로 일곱줄의 여백을 주어 1페이지당 55줄 단위로 프린트하게 되어 있다. 이와 같은 기능으로는 노턴 유틸리티에 있는 LINEPRNT.COM이 있다.

〔표 2〕 색상코드

코 드	색 상	코 드	색 상
0	흑 색	8	회색
1	청 색	9	밝은 청색
2	녹 색	A	밝은 녹색
3	청 색	B	밝은 청색
4	적 색	C	밝은 적색
5	자주색	D	밝은 자주색
6	갈 색	E	노란색
7	백 색	F	밝은 백색

이 두개의 프로그램은 큰 차이점은 없지만 약간의 차이점은 갖는다. 즉 LINEPRNT. COM은 헤더를 한 라인에 출력하지만 PR.COM은 앞부분에 상자를 만들어 그곳에 화일의 이름, 페이지 수, 그리고 그 화일이 작성된 날짜와 시간이(LINEPRNT.COM은 출력하는 현재 날짜와 시간으로 출력됨) 모두 기입되어 화일관리에 도움을 준다는 점이 그것이다. 실행 예를 화면 2에 보였다.

RENDIR.COM

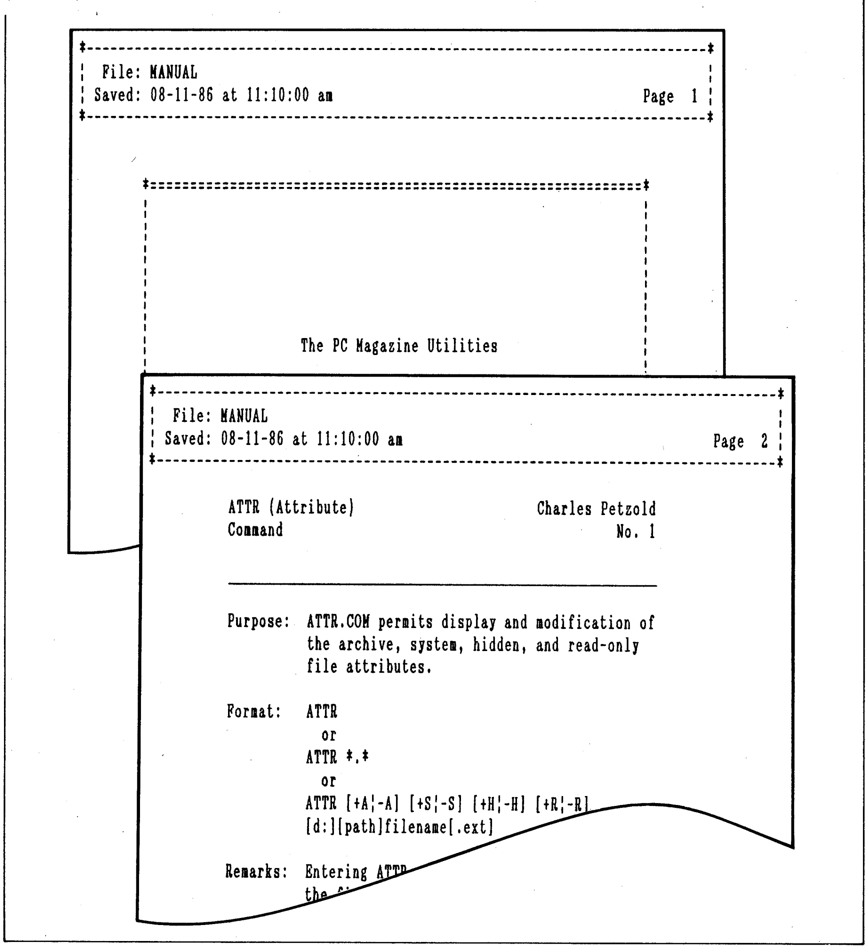
서브 디렉토리의 이름을 변경하는 프로그램이다. 컴퓨터를 사용하는 사람이라면 RENAME이라는 DOS 내부 명령어를 모르는 사람은 없을 것이다. 이것은 기존의 화일이름을 다른 이름으로 변경하고자 할 때 사용하는 명령이지만 문제는 이것이 애석하게도 서브디렉토리의 이름까지는 지원을 해주지 못한다는 것이다.

서브디렉토리는 플로피디스크에서는 많이 이용되지 않으나 하드디스크를 사용할 경우는 꼭 필요한 것이다. 또 보통의 경우 서브디렉토리의 이름을 변경할 경우가 종종 있는 것이다.

이 유틸리티의 사용은 DOS의 RENAME 과 동일한 방법으로 사용된다. (단 버전 3.0이상의 DOS에서만 실행가능)

```
A:\>RENDIR C:\PAS PASCAL4
```

〔화면 2〕 PR.COM의 실행 예


```
File: MANUAL
Saved: 08-11-86 at 11:10:00 am                    Page  1

        The PC Magazine Utilities

File: MANUAL
Saved: 08-11-86 at 11:10:00 am                    Page  2

ATTR (Attribute)                  Charles Petzold
Command                                     No. 1

Purpose:  ATTR.COM permits display and modification of
          the archive, system, hidden, and read-only
          file attributes.

Format:   ATTR
            or
          ATTR *.*
            or
          ATTR [+A¦-A] [+S¦-S] [+H¦-H] [+R¦-R]
          [d:][path]filename[.ext]

Remarks:  Entering ATTR
          the fi
```

지금까지 여섯가지의 유틸리티만 소개를 하였는데 KEY-FAKE.COM이나 BROWSE.COM등은 독자들도 유용하게 쓰일 수 있는 유틸리티라고 본다.

이상으로 PC-매거진의 간단한 소개를 끝마친다. 이밖에도 PC-매거진과 같은 공개 소프트웨어는 많이 있다. 우리나라에서도 많은 공개 소프트웨어가 나왔으면 하는 바램이고 부분적으로 현재 PC통신을 통해서 공개 소프트웨어가 교환되며 활용되고 있다.

바코드시스템

박진원/삼미전산 바코드사업부

바코드의 흐름을 알기 위해서는 바코드의 시조라 할 수 있는 미국을 예로 들지 않을 수 없다. 저마다 다른 특성을 내포하고 있는 그들의 각 산업분야에서 바코드는 어떠한 방법으로 도입·활용되어왔는지를 살펴보는 것이 순서일 것 같다.

미국내의 거의 모든 생산품에서 발견할 수 있는 바코드 라벨은 지난 20년에 걸친 생산품 분배의 자동화를 이룩한 중요한 진보를 상징하고 있다.

오늘날 슈퍼마켓이나 백화점등에서 엄청나게 다양한 생산품들이 무리없이 취급될 수 있는 것은 생산, 포장, 취급등에 적용된 자동화 시스템의 결과라고 할 수 있다.

이러한 바코드 체제를 응용한 자동화 시스템의 혁신은 하룻밤사이에 일어난 것은 절대 아니다. 그것은 식료품 소매산업을 통한 자동화기술의 꾸준한 응용의 결과였다.

여기에는 1950년대에 나타난 트랜지스터의 발명으로 인한 전자공학의 획기적 진보가 주요 요인으로 작용했음을 간과할 수 없다. 이러한 몇몇 요소가 판매처리에 대한 자동화를 가속시켰으며, 바코드가 매력적인 데이터 처리방법으로 인정받기 시작하면서 응용 및 개발이 가속화되는 계기가 되었다.

근래에 와서는 매장 관리, 재고 관리, 사무 자동화, 공장 자동화 등 거의 모든 분야에 폭넓게 적용되고 있다.

바코드의 구성

거의 모든 바코드는 다음과 같은 구성으로 이루어져 있다(그림 1 참조).

●Quiet Zone

바코드 검색기의 정확한 판독을 위해 설명되는 부위로, 바코드 부호의 전후에 위치하여 통상 "X" 디멘젼 10배 크기의 영역이다.

●시작문자 및 종료문자

바코드로 표현하고자 하는 데이터의 앞/뒤에 각기 위치하며, 바코드의 종류에 따라서 내용 및 구성이 다르다. 바코드가 검색기에 의해서 검색될 때 검색의 방향을 인지할 수 있게 해줌으로써 바코드의 양방향성 검색을 가능하게 해준다.

●바와 여백

바코드 데이터는 폭이 넓은 바, 가는 바, 그리고 넓은 여백, 좁은 여백의 4가지가 서로 조합되어 표현된다. 이때, 이 요소 하나 하나를 엘리먼트(Element) 라 부르며, 4개의 엘리먼트 중에 좁은 성분과 넓은 성분은 반드시 일정한 비율이 필요하다(1:2, 1:2.5, 1:3 등의 비율로 구성된다.).

●"X" 디멘젼

바코드 데이터를 구성하는 요소중에 좁은 성분 즉 가는 바 및 좁은 여백의 폭의 크기를 "X" 디멘젼이라고 부른다.

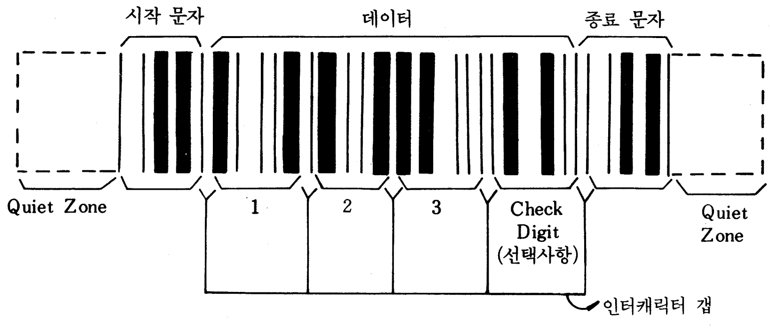
●인터캐릭터(Intercharacter) 갭

바코드 데이터를 구성하는 개개의 문자를 구별하기 위해 각 문자사이에 설정하는 영역으로, 크기는 "X"디멘젼과 같으며 이 갭의 유/무에 따라 모든 바코드는 연속코드 및 비연속코드로 구별된다.

●인터프리테이션(Interpretation) 라인

바코드 데이터의 아래나 위에 그 바코드의 내용을 사람이 육안으로 식별할 수 있도록 나타내는 것을 말하며, 이것은 사용자의 필요에 의존한다.

〔그림 1〕 바코드의 구성



바코드의 특성

●바코드의 정보밀도

바코드를 구성하는 바와 여백의 굵기는 바코드 검색기(Scanner)의 성능과 밀접한 관계가 있으며, 일반적인 바코드에서 가는 바의 폭의 크기는 약 0.25㎜이다. 또 가는 바의 폭과 굵은 바의 폭과의 비를 배율이라고 하는데 이를 식으로 정리해 보면 다음과 같다(그림 2 참조).

바코드의 규격을 표시할 때에는 주로 좁은 바의 폭과 배율로서 나타내는 경우가 많으며, 가는 바의 폭이 좁아질수록 바코드 검색기는 고성능화되어야 판독이 가능하다.

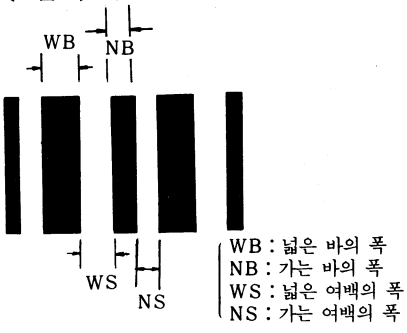
●바코드와 칼라와의 관계

대부분의 제품에 붙어 있는 바코드 표식도 다양한 색상으로 인쇄되어져 있는 것을 볼 수 있을 것이다. 얼마전까지만 하더라도 이러한 색상이 처리된 바코드의 대부분은 검색되지 못하였으나, 광학 기술의 발달에 힘입어 최근에는 다양한 색상들이 처리된 바코드도 모두 판독이 가능하게 되었다. 그러면 바코드와 색상과는 어떠한 연관이 있으며, 또 바코드가 색상에 의해 어떠한 제한을 받는지 알아보기로 하자.

(1) 빛과 색의 특성

빛은 복사에너지의 일부로 구분되며

〔그림 2〕 바코드의 정보밀도



$$배율=\frac{넓은\ 바의\ 폭}{가는\ 바의\ 폭}=\frac{WB}{NB}=\frac{WS}{NS}$$

각각의 파장대를 가지고 있다. 이중 특정 파장대의 빛만을 눈으로 볼 수 있으며, 이를 가시광선이라고 부르고 있다. 이러한 빛을 물체가 흡수하거나 반사하게 되면, 이것이 곧 그 물체의 색이 된다. 즉 어떤 물체가 모든 가시광선의 파장을 반사할 경우, 이 물체의 색을 흰색으로, 반대로 모두 흡수할 경우 이를 검정색이라고 부르고 있다. 그러나 색상 처리된 바코드에 비치는 광선이 가시광선이 아닌 경우는 광 검색기가 인쇄된 바와 바탕사이의 대조를 감지할 수 없게 되며, 색상 처리가 무의미해진다. 따라서 이와같은 광원(Light Source)에 제약을 안받기 위해서는 몇가지 대안이 필요한데 그 중의 하나로 감열지에 바코드를 인쇄하는 방법이 있다.

(2) 바코드의 색상 및 광원

광원으로는 태양빛, 성냥불, 전구빛, 레이저, 전기적으로 도체화된 반도체 광선 혹은 LED 등 여러가지가 사용되어질 수 있다.

이중 바코드에 중요한 광원을 검토해 보기로 한다.

① LED

접촉 방식의 검색기인 원드(Wand)의 광원으로 주로 사용되고 있으며, 최근에는 CCD 센서를 이용하여 데이터를 한꺼번에 검색하는 방식, 비접촉 방식, 셀프 스캐닝(Self Scanning) 방식 등의 새로운 모델들이 이를 광원으로 채택하고 있다.

레이저방식에 비해 판독률이 상대적으로 떨어지나 사용하기 쉽고 가격이 저렴하기 때문에 널리 이용되고 있다. LED는 광원별로 가시광선 LED(630~680㎜의 빨간색)와 적외선 LED(820~900㎜)의 2가지로 나누는데 가시광선 LED는 뒤에 설명한 He-Ne 레이저와 같은 특성을 나타내게 되며, 적외선 LED는 탄소성분 함유율이 낮은 바코드는 판독하지 못하는 단점이 있으나 이 특성을 이용하여 보안처리가 필요한 곳에 이용되기도 한다.

② He-Ne 레이저

비접촉 방식의 검색기에 주로 쓰이며, 5㎝에서 수 미터까지 판독이 가능하고, 짧은 거리에서 사용할 경우 고밀도의 바코드를 판독할 수 있어 슈퍼마켓, 백화점, 공장등에서 널리 사용되고 있다. 또한 초당 300~900회까지 판독 가능한 고속 셀프 스캐닝 방식도 있어 고속 컨베이어같은 것 등에 적용할 수 있다. LED형에 비해 외형이 크고 가격이 고가이며, 수명이 짧은 것이 단점이다.

빨간색의 가시광선을 내보내며, 바가 빨간색으로 인쇄되어 있는 경우(이 경우에는 바탕으로 부터도 역시 빨간색이 반사되므로 바와 여백사이의 대비가 이루어지지 않아 검색이 불가능해짐)를 제외하고는 다른 다양한 색상으로 바코드를 인쇄하는 것이 가능해진다.

③ 레이저 다이오드(LD)

He-Ne 레이저의 단점을 보완할 수 있는 형으로, 전력소비가 적고 수명이 길며 가격이 낮아 근래에 많이 이용되고 있다. 다만 He-Ne 레이저에 비하여 사용 거리가 짧고, 파장이 적외선 영역이기 때문에 적외선 LED가 가진 단점을 내포하고 있기는 하지만, 조만간 가시광선 LD의 상용화가 예상되어 그 사용영역이 확대되리라 본다.

결과적으로, 바코드는 바와 바탕사이의 대비를 인식할 수 있는 검색기가 존재할 때 의미가 있으며, 또한 바코드의 인쇄방법을 설정하기 전에 사용자는 바코드를 검색시 사용할 검색기 광원의 파장을 미리 알아야 한다는 점을 기억해야 한다.

일반적으로 바는 검정색, 파란색, 초록색, 자주색 등이 쓰이며, 바탕색으로는 흰색, 노란색, 주황색, 빨간색 등이 쓰이는데, 빨간색의 경우는 바코드 복사방지의 일환으로 사용된다.

바코드의 종류

바코드의 종류는 수십가지가 있지만 여기서는 널리 쓰이는 주요한 바코드의 예를 설명하겠다.

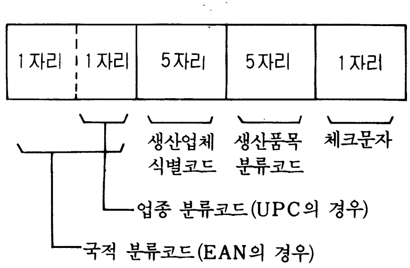
●UPC / EAN

유통 및 소매업의 POS(Point of Sales) 체제에서 공통 상품코드로 널리 사용되고 있으며, 1973년 미국, 캐나다의 식품업체에서 채택되어 북미에서 널리 사용중인 UPC(Universal Product Code)와 1977년 유럽에서 UPC를 변형하여 규격화시킨 EAN(European Article Numbering System)의 2개 코드는 우리 일상생활과 상당히 친숙한 관계에 있는 코드이다.

일본에서는 EAN의 일종인 JAN을 1978년에 JIS(일본공업표준규격) 규격으로 제정하였으며, 우리나라도 1988년 5월에 제14차 EAN 총회에서 정회원으로 가입하여, "880"이라는 고유 국가식별 코드를 부여 받았다.

그림 3에 UPC/END코드를 비교하였다. UPC는 보다 보편적이고 광범위한 EAN의 한 부분이라고 할 수 있으며, EAN을 해독하는 판독기로 UPC를 해독할 수는 있으나 그 역으로는 해독할 수 없다. 따라서 이러한 점은 바코드체제 도입시 고려해야할 사항임을 명심해야 한다. 또한, 공통 상품코드에 있어서 숫자(0~9)만을 표시할 수 있으며, 마지막 자리에 데이터 판독의 오류 여부를 확인하기 위한 체크 문자를 두어 데이터의 신뢰성 향상을 꾀하고 있다.

〔그림 3〕 UPC/EAN코드의 비교



●코드 39

영문 데이터의 바코드 처리에 대한 요구를 충족시키기 위해 개발된 코드로, 모듈(Module) 43이라는 자체 검사법을 도입하여 가장 신뢰도가 높은 코드로 알려져 있다. 3 of 9 코드라고도 불리우며, 이는 각 문자를 구성하는 9개의 성분중에 3개가 넓은 요소(바 혹은 여백)로 이루어졌기 때문이다.

●2 of 5 코드

1968년에 세계 최초로 개발된 코드로서 현존하는 코드중 가장 간단한 형식을 취하고 있으며, 인더스트리얼(Industrial) 2 of 5 코드라고도 불리운다. 이 코드의 부류로 이밖에 인터리브드(Interleaved) 2 of 5 코드 및 매트릭스 2 of 5 코드가 있으며, 표시 가능한 글자는 숫자뿐이며 크기의 변화가 가능하다.

인더스트리얼 2 of 5 코드는 최초의 바코드로, 이후 개발된 바코드 체제의 기본이 되었으며, 2 of 5라고 부르는 이유는 각 문자가 5개의 바로 구성되는데 이중 2개는 넓고 3개는 좁기 때문이다.

이 코드의 특징은 오직 바로만 구성되다는 점이다.

인터리브드 2 of 5 코드는 2 of 5 코드 부류중에 가장 높은 밀도를 유지할 수 있는 코드로 셀프 테스트능력 또한 뛰어나 산업용 뿐만 아니라 기타 다른 여러 분야에 걸쳐 널리 이용되고 있다. 이 코드에서는 각 바 사이의 여백도 글자의 구성 요소로 작용하여, 홀수번째 문자는 바로, 짝수번째 문자는 여백으로 하여, 서로 삽입한 형태로 구성이 되어 하나의 패턴을 취하기 때문에 데이터의 수는 반드시 짝수이어야 하며, 홀수일 경우에는 선두에 0을 강제 삽입하여 짝수 처리를 한다. 이 코드의 이름은 1문자가 5개의 요소로 구성되는데 이중 2개가 넓은 요소(바 또는 여백)인데서 유추된 것이다. 각 문자간 갭은 없는 셈이다.

매트릭스 2 of 5 코드는 정보의 표현 밀도가 높고 각 문자사이의 여백은 정보를 포함하지 않으므로 판독시 오류 발생률이 적다. 이 코드는 3개의 바와 2개의 여백으로 1문자를 나타내는 독특한 구성 방식을 취하고 있으며, 5개의 요소중 2개는 넓고 3개는 좁게 구성된다.

●코더바(Codabar)

1972년 모너크 마킹 시스템(Monarch Marking System)에서 발표한 코드로, 각 문자는 7개의 요소(4개의 바, 3개의 여백)로 구성되는데 이중 2개가 넓은 요소이기 때문에 2 of 7 코드라고도 불리운다. 숫자 및 6개의 특수기호 등 총 16개의 글자를 표현할 수 있으며, 영문이 필요하지 않으면서 가변을 요하는 곳, 특히 혈액원, 도서관, DPE(사진 현상)의 봉투등에 널리 사용되고 있는 코드이다.

바코드의 해독원리

바코드 데이터를 판독하는 장치는 다양하게 분류될 수 있으나, 해독원리는 거의 동일한 방식으로 되어 있다.

결국, 바코드에 빛을 주사시켜 반사되어 나오는 반사광을 검출·증폭하여 정보를 판독하는 방식인 것이다. 이러한 판독체제는 검색기의 광원부 및 검출부, 판독기의 증폭부, 신호변환부, 해독부 및 데이터 전송부로 구성된다.

●광원부(Light Source)

첨단 전자기술을 응용한 다양한 소자들이 광원으로 쓰이고 있으며, 이중에서도 LED, He-Ne 레이저, LD 등이 주로 쓰이고 있다. 이들은 각 소자별로 각기 다른 파장대의 빛을 방출한다.

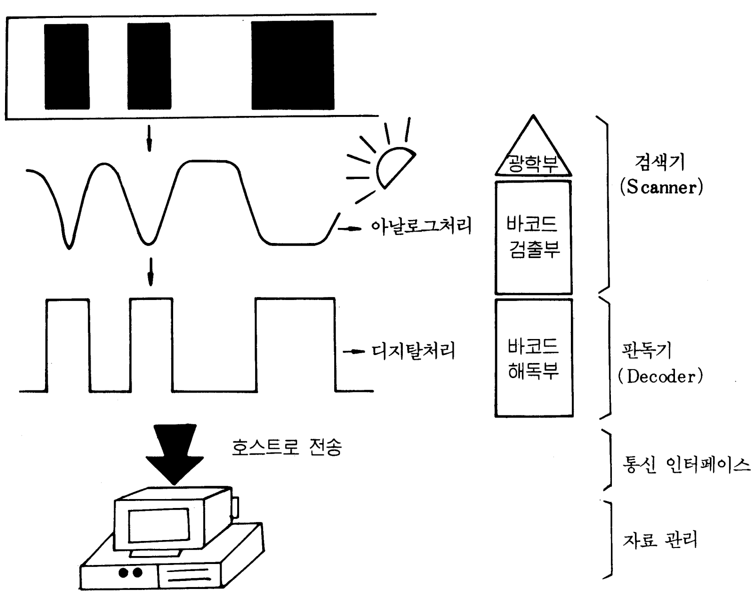
●검출부(Photo-Detector)

광원으로 부터 나온 빛은 바코드와 여백부분에서 각기 다른 반사율을 나

〔그림 4〕 바코드의 종류



〔그림 5〕 바코드 해독 원리



타낸다. 광검출부는 이러한 반사광을 받아 반사율의 차이를 이용해 이를 전기적 신호(아날로그방식)로 바꿔준다.

●증폭부(아날로그 증폭기)

광검출부에서 출력된 아날로그 신호(바코드 데이터)는 후속 처리를 위해 아날로그 증폭기에서 증폭이 된다.

●신호 변환기(A / D 콘버터)

논리적인 처리(판독)를 위해 아날로그 신호를 디지탈 신호로 바꿔준다.

●해독부(Decoder)

"1"과 "0 으로 구성되는 디지탈 신호를 미리 정해진 심볼(영문, 숫자, 기호…)로 변환시킨다.

●전송부

변환된 심볼은 ASCII 코드 값으로 구성되며, 이를 호스트로 전송, 데이터 처리를 가능케 한다(그림 5 참조).

바코드 디바이스 종류와 특성

바코드 디바이스는 크게 검색기, 판독기, 집중기(Concentrator), 프린터 등으로 구분할 수 있다.

●검색기(Scanner)

검색기는 운용형태에 따라 접촉식과 비접촉식으로 나눌 수 있으며, 접촉식의 대표적인 것으로 펜형과 CCD 형이 있으며, 비접촉식은 휴대형 건(Gun)과 스캐너로 나눠진다.

(1) 원드 스캐너

핸드 헬드 라이트 펜(Hand Held Light Pen)은 원드(Wand) 스캐너라고도 불리우며 낮은 가격과 휴대가 간편한 특성으로 인해 가장 많이 사용되고 있다. 검색원리는 사용자가 심볼을 가로질러 펜을 움직이면 바와 여백사이의 반사율차를 포토 리셉터가 감지하여, 이 반사율의 차를 아날로그 신호로 변환하여 디코더로 보내게 된다(그림 6, 7 참조).

(2) CCD 스캐너

터치 스캐너라고도 불리우며 해독방식은 라이트 펜과 비슷하나 한개의 LED를 광원으로 사용하는 라인트 펜에 비해, CCD(Charge Coupled Device)

스캐너는 여러개의 LED를 사용하는 CCD 이미지 센서로 처리한다는 점이 다르다. 이것은 번짐상태의 바코드나 비닐, 셀로판같은 곡면이나 요철면상의 바코드를 높은 정밀도로 판독할 수 있게 하며 가동부가 없어서 안정성, 고신뢰성을 가진다. 그러나 멀리 떨어진 바코드나 판독 구경보다 긴 바코드는 판독이 불가능하다는 단점이 있다. 일반적으로 50~100㎜까지 판독이 가능한 제품이 소개되고 있다(그림 8 참조).

(3) 건(Gun) 스캐너

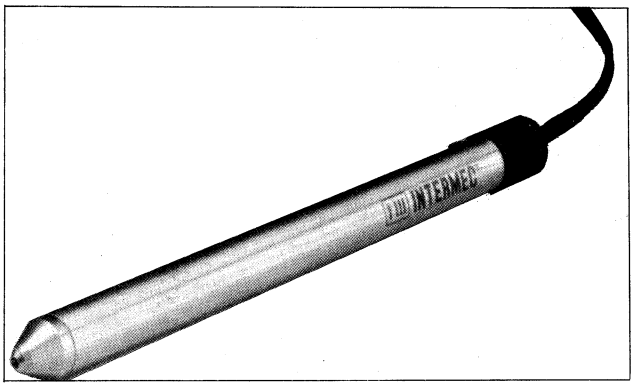
비접촉 방식으로, 휴대 및 운용이 간편하고 데이터 입력처리가 신속하며, 곡면이나 탄력성이 있는 곳에 부착된 심볼의 검색이 가능하다. 또한 레이저 다이오드 및 He-Ne 레이저기술을 이용하여 광학 검색 특성이 우수한 반면 펜형보다 가격이 비싸다는 단점이 있다(그림 9 참조).

(4) 고정형 검색기(Fixed-Mount Beam Scanner)

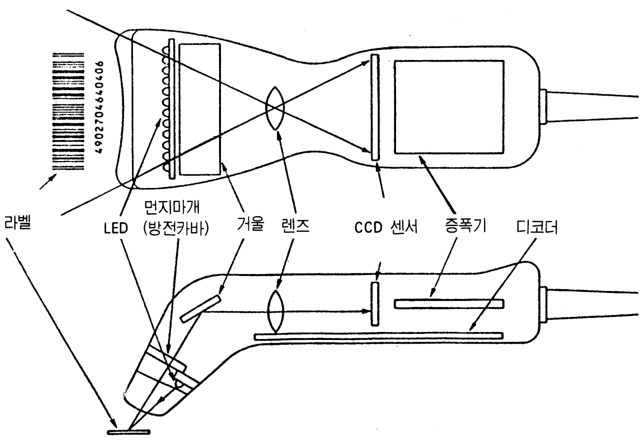
빔의 주사 형태에 따라 정적(Fixed)빔 주사방식과 동적(Moving)빔 주사방식이 있으며, 빔의 주사방향을 고정시키고 바코드 심볼의 이동속도를 이용하여 검색하는 방식이다.

동적(Moving)빔 주사방식이 요즘 많이 쓰이고 있다. 동적빔 주사방식은 보통 400~900회/초의 고속 검색을 함으로써 한개의 바코드를 여러번 읽어 판독오류에 대한 신뢰도를 향상시킬 뿐 아니라 바코드 심볼의 이동속도에 유연성을 가질 수 있도록 다양한 모델이 나와 있다. 광원은 반도체 레이저와 He-Ne 레이저가 주로 쓰이며, 반도체방식은 He-Ne방식보다 상대적으로 크기가 작고 가격이 낮으나 OR(Operating Range)과 DOF(Depth Of Field)가 짧고 검색속도가 떨어진다. 최근에는 3차원 형태의 레이저 검색기가 개발되어 바코드가 포장박스의 어느 부분에 부착되더라도 판독이 가능하게 되었다. 통상 이러한 동적빔주

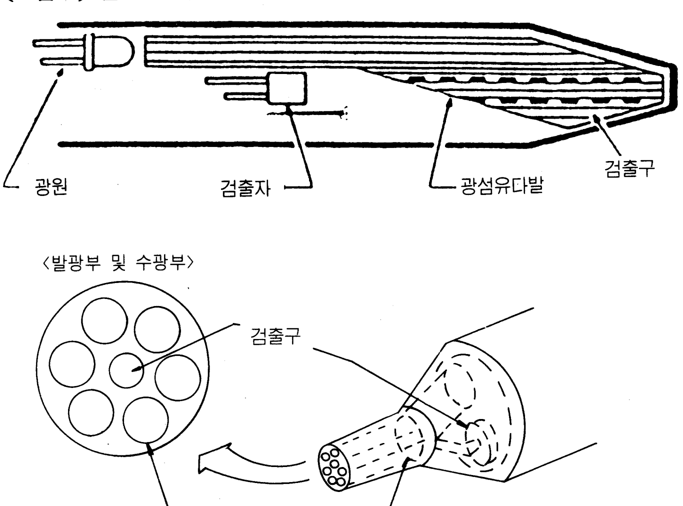
〔그림 6〕 원드(Wand) 스캐너



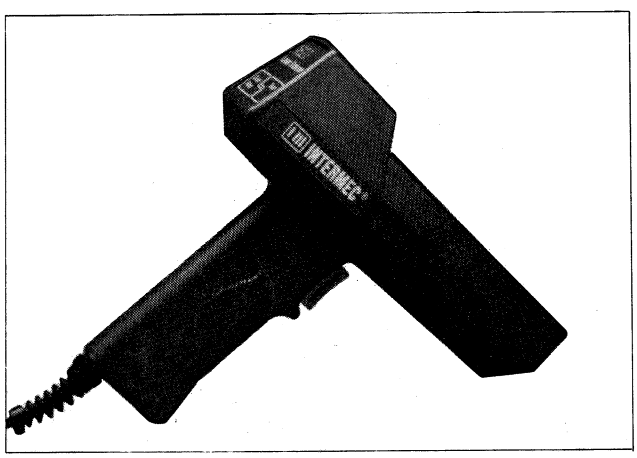
〔그림 8〕 CCD 스캐너의 내부구조



〔그림 7〕 원드 스캐너의 내부구조



〔그림 9〕 건(Gun) 타이프 스캐너



사방식은 검색과 판독기능을 같이 갖추고 있다.

이외에도 IAR(Imaging Array Reader) 라는 카메라의 기능을 본딴 모델과 뱃지(Badge) 스캐너가 있다(그림 9, 10 참조).

●판독기(Decoder)

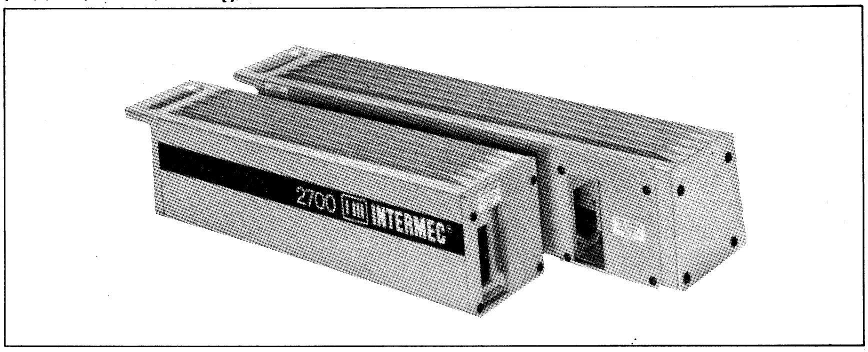
검색기에서 보내온 아날로그 신호를 디지탈신호로 변환하고, 이를 다시 컴퓨터가 처리할 수 있는 ASCII 코드값으로 변환시키는 기능을 갖고 있다. 그림 11 시스템은 구성방식에 따라 온라인형과 배치처리형으로 구분되며, 온라인형은 생산공정 라인같은 곳에 설치되어, 발생하는 데이터를 실시간 처리로 호스트에 전송하는 경우에 주로 쓰이며, 또 온라인방식과 같은 개념이나, 컴퓨터 본체와 키보드사이에 판독기가 위치하여 일반적인 데이터 입력방식(키입력 작업)과 바코드 입력방식을 동시에 쓸 수 있도록 한 키보드 에뮬레이션(혹은 Wedge)방식도 있다.

배치처리형은 컴퓨터를 직접 설치할 수 없는 환경에서, 데이터를 수집한 후 이를 유선 혹은 무선으로 호스트에 전송하는 경우에 쓰이며, 휴대형 판독기가 그것이다.

●집중기(Concentrator)

여러개의 바코드 디바이스를 효율적으로 운용하기 위한 장비로서 미국 인터멕(Intermec)사의 9161B모델을 예로 들면, 최대 140대의 장비를 연결·처리할 수 있으며, 각 폴드(Polled) 장치로부터 전송된 데이터를 필요시 저장할 수 있도록 5만바이트의 메모리를 내장하고 있다. 또한 벌크 스토리지(Bulk Storage)를 이용하여 메모리의 외부확장이 가능하며, 터미널을 연결하여 집중기와 호스트간에 주고 받는 메시지 및 코맨드를 디스플레이시켜 데이터의 전송 상황을 추적할 수 있는 기능도 갖고 있다. 또한 여러대의 장비를 연결하여 데이터를 받아들일 수도 있다.

〔그림 10〕 동적(Moving) 빔 주사방식 스캐너



●프린터

바코드 인쇄는 판독 효율과 직결되어 바코드 시스템의 성공을 가름하는 열쇠가 되고 있다. 일반적인 인쇄 방법은 오프사이트프린팅(Off-Site Printing) 및 온사이트 프린팅(On-Site Printing)의 두가지 방법으로 구분되며, 바코드의 인쇄는 두가지 모두 가능하다. 바코드 인쇄시 고려해야 할 사항은, 인쇄된 바코드의 “X” 요소와 판독기의 발광단자 규격이 일치하지 않는다던가 혹은 판독기 광원의 파장이 인쇄된 바와 바탕과의 대비를 충분히 처리할 수 없는 경우는 판독오류가 발생하게 된다. 이러한 판독오류는 인쇄된 라벨과 판독기 모두에서 발생할 수 있는 문제로 바코드 인쇄시에는 먼저, 사용할 판독기의 성능 및 규격을 충분히 고려할 필요가 있다. 결국 바람직한 바코드 인쇄기법의 선정은 첫째, 사용자가 사용하기 편리하고, 둘째, 신뢰할 만한 바코드를 선택한 후 판독기의 효율을 고려하여 적합한 프린터를 선택하는 것으로 이루어져야 할 것이다.

바코드 인터페이스

바코드 인터페이스 구성방식은 현재의 업무 내용에 좌우되는 성질의 것이며, 호스트의 종류, 업무 처리내용, 사용될 바코드 장비의 수량 및 현장의 운용 환경에 따라 시스템 구성방식이 제시될 것이다.

●구성내용에 따른 구분

(1) 크로스바 데이터 콜렉션시스템

주컴퓨터 혹은 PC 레벨의 호스트와 하위 바코드 장치들의 중간에 집중기(Concentrator)를 설치하여, 토탈(Total) 개념의 바코드 체제를 구성하는 경우이다. 여기서의 집중기는 M개의 출력회선을 그보다 작은 수인 N개의 입력회선으로 집중화하는 장비로서, 하나의 호스트로 복수개의 수많은 바코드 장치들을 통제할 수 있도록 해준다. 이러한 집중기는 상당히 중요한 의미를 갖고 있기 때문에 별도의 부문으로 다뤄서 설명하고자 한다.

(2) Stand-Alone 구성

PC나 인텔리전트, 더미(Dummy) 터미널에 바코드 장비들을 1대 1로 접속시키는 구성이며, 비교적 업무처리 내용이 단순한 경우에 적용되는 방식이다.

●통신방식에 따른 검토

호스트와 바코드 장치간의 통신방식은 통상, 유선방식에 의해 이루어지나, 업무내용이나 현장 환경에 따라 무선방식(RF 시스템)을 쓰는 경우도 있다. 또 호스트와 디바이스간의 거리에 따라 라인 드라이버(RS-232C, RS-422, RS-485 멀티 드롭, 20mA 전류 루프, 데이지 체인(Daisy Chain), 모뎀, 어쿠스틱 커플러(Acoustic Copler)의 모델이 결정되며, 이외의 특수한 예로, 호스트의 옵션 통신포트를 이용하는

방식이 아닌 키보드 에뮬레이터(Wedge) 방식이 있다.

●집중기(Concentrator)

콘트롤러라고도 하며, 앞에서 언급했듯이 하나의 호스트로 복수개의 바코드 장비를 통제하는 기능과 IBM 호스트의 BSC / SDLC, EBCDIC방식을 ASC, ASCII방식으로 변환하는 기능으로 구분할 수 있다.

집중기에 접속되는 바코드 단위가 복수대일 경우, Point-To-Point방식과 Multi-Drop 방식으로 구성되는 것이 일반적인 형태이며, 멀티 드롭의 경우 각 바코드 단말기에 폴링(Polling)할 수 있는 시스템이어야 하며, 집중기는 반송하는 형태로 한 회선에 연결된 모든 단말기에 블록 형태의 어드레스 데이터를 전송하면, 바코드 단말은 블록데이터 어드레스와 자신의 어드레스를 비교하여 어드레스가 일치된 단말만 데이터를 송신하게 되며, 따라서 각 단말은 어드레스 판단 기능과 데이터 블록을 일시적으로 저장할 수 있는 버퍼 메모리를 가지고 있어야 한다.

그러나 온라인 개념의 처리체제에서는 이러한 방식과는 달리, 각 바코드 단말에서 데이터가 리얼 타임으로 수집되어 집중기로 전송되어 오면, 집중기에서는 수신 포트를 인식하여, 모아진 각 데이터에 어드레스를 붙여 호스트로 전송하여 처리하게끔 하는 방식을 많이 쓰고 있다.

이러한 멀티 드롭방식은 회선 사용료를 절감할 수 있으나 회선 고장시 고장지점 이하의 하위 단말은 모두 운용불능이 되어 버리는 단점이 있다. 이러한 방식에서는 폴링과 셀렉션의 개념이 중요한 의미를 갖고 있으며, 각 메이커별, 디바이스별로 다양한 프로토콜(Protocol)이 제공되고 있다.

또, 로컬 네트워크인 경우 거리에 따라 RS-232(15M), RS-485(600M), RS-422(1.2kM)방식을 선택, 적용하면 된다.

바코드 시스템의 응용

바코드를 응용하여 시스템을 구축한다는 것은 현재 수작업으로 자료를 수집, 컴퓨터에 입력시키는 모든 곳곳에서 자료입력을 정보 발생시점 즉 리얼타임(Real Time)에 접근시키고, 높은 신뢰도를 갖도록 하여 호스트 컴퓨터와 보다 원활하고 신속한 통신을 가능케함으로써 높은 부가가치를 갖는 경영정보를 창출해내는 것을 의미하여 그 영역은 무한히 넓지만 여기서는 공장 자동화(FA)분야, 유통분야, 일반 사무자동화 업무(OA) 및 기타 서비스 업무 분야로 나눠 살펴보겠다.

●공장 자동화와 POP

공장 자동화의 변천을 살펴보면 초기의 NC 공작기계, 산업용 로보트 등의 개별 자동화 개념에서 CAD / CAM, 자동 창고, 반송 미니 컴퓨터 등의 주변기기의 도입으로 인한 토탈 자동화 체제의 개념으로 바뀌고 있다. 이런 상황에서 바코드 관련 기술이 정보의 비접촉 판독가능, 고속입력, 고신뢰성과 더불어 H / W, S / W 양면에서 현장의 여러가지 요구사항에 부합될 수 있다는 특징으로 인해 공장자동화 현장에 POP 시스템이 대량 도입될 추세이다.

POP(Point Of Prodution)시스템은 생산활동이 행해질 때마다 발생하는 정보를 그 시점에서 포착하여 관리 및 경영상에 필요한 각종 정보로 가공하여 제공하는 체제를 말한다. 즉 공장 등에서 생산현장의 기계, 설비, 작업자의 업무 등으로부터 실시간으로 수집된 생산 시점의 관리정보를 가공처리하여 출하관리, 공정관리, 생산계획 등에 환원시키는 것으로, 공장의 경영 합리화를 위해서 필수적인 유연성과 생산성을 동시에 높일 수 있는 시간적 개념의 최적화를 지향하고 있음으로 해서 POP체제는 시대적 요구사항이라고 까지 평가되고 있다.

●유통 분야와 POS

최근 경제, 사회 전반에 걸쳐 정보의 중요성이 크게 대두되고 있는 시점에서 유통산업도 예외는 아니어서 경영의 합리화, 근대화를 추진하거나 소비자 요구를 정확히 파악할 목적으로 POS 체제(Point of Sales : 판매시점정보관리)나 온라인 수 · 발주 체제를 구축하는 등 정보의 시스템화가 급속히 추진되고 있다. 이에 따라 각국 정부는 POS 체제의 도입 초기부터 각종 상품코드의 표준화, POS체제의 활용 연구, POS 도입에 관한 융자제도의 신설 등 POS 체제의 보급을 위한 환경정비에 많은 노력을 기울여왔다. 국내에서는 현재 슈퍼마켓, 백화점, 유통업계를 비롯, 호텔 등과 같은 곳에서 이 시스템이 적지않게 이용되고 있으나 아직은 극히 초보적인 단계에 머물고 있으며, 대한상공회의소에서도 공통 상품코드 계획 등 5개년 기간별 계획을 수립하여 그 기반 확립에 힘쓰고 있다. POS에 대한 정의는 관점에 따라 다소 차이가 있으나, 대체로 광학 판독장치에 의해 품목별로 수집된 판매정보, 매입, 발주 및 운송 등의 단계에서 발생하는 각종 정보를 컴퓨터를 이용하여 각 부문의 전략적 의사결정정보로 활용될 수 있도록 유효하게 처리, 가공, 전달하는 시스템이라 할 수 있다.

●OA업무 및 기타 서비스

사무자동화 분야의 주요 도입사례를 보면,

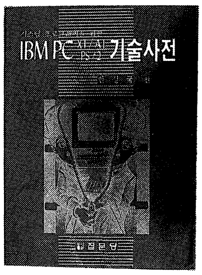
-자산 및 문서관리
-근태관리
-출입, 보안관리
-도서관리
-의료기관의 병원업무관리

등 이밖에 품질관리, 재고관리, 제품포장 및 출하관리 등에 적용할 수 있으며, 이러한 몇가지 부분들을 통합관리할 수 있는 패키지로 나와 있다. PC

신간안내

IBM PC XT/AT/PS/2 기술사전

한성국 편역/집문당 발행/
정가 10,000원/776면/4·6배판



지금까지 IBM-PC의 시스템에 관한 내용을 담은 책은 거의 전무하다시피 하였다. 그런 상황에서 이 책은 IBM-PC의 시스템에 관한 정보를 수록하여 시스템 프로그래머에게는 좋은 자료로서의 역할을 할 수 있을 것이다.

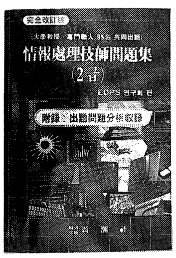
이책은 IBM-PC에서 사용되는 데이터와 코드형태, 그리고 MS-DOS 관련 소프트웨어부분으로는 명령어의 요약과 실행화일 구조 등을설명하고 일반적으로 정확히 이용하지 못하는 DOS 함수콜을 기능별로 찾아서 이용할 수 있도록 목차부터 구별하였다. 또한 시스템을 제어하는 BIOS정보를 일목요연하게 정리하여 시스템개발자가 이용하기 편리하게 구성하였다.

DOS인터럽트, 마우스, 확장메모리에 대해서도 상세히 다루고 있으며, CGA, EGA, VGA 등에 관한 정보도 다양하고 포괄적으로 제공하고 있다. 하드웨어 관련 정보도 다른 어느 서적보다 다양하게 취급한 일종의 기술사전이라고 말할 수 있다.

이책은 IBM-PC 계열의 기술을 집약하여 수시로 참고할 수 있게 구성하였기 때문에 시스템 프로그래머나 시스템에 대하여 공부하려는 학생들에게 필독서가 될 수 있을 것이다.

정보처리기사 문제집(2급)

EDPS 연구회편/상조사 발행/
정가 6,500원/454면/4·6배판



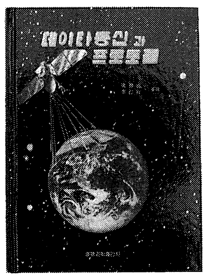
국가공인기술 자격제도에서 전문성과 기술적인 수준향상을 위하여 2차시험에서 실기문제를 다루도록 하고 있다.

이책은 이러한 제도적변화 특성에 맞추어 정보처리기사 2급을 대비하는 응시자들을 위하여 출제 유경험자를 포함한 각계의 전문가 및 대학교수를 중심으로 문제 은행식 출제를 하여 기사를 대비하는 사람들에게 좋은 참고서가 되리라 본다.

전자계산기 일반, 계산기 구조, 어셈블리언어 일반, 프로그래밍 언어, 자료구조, 기초통계 일반 및 응용 등 각 부분을 출제경향별로 구분하여 수록하였고 부록에서는 지금까지의 기사시험 출제경향을 분석하고 있다.

데이타통신과 프로토콜

성승희·이인형 공저/
홍능과학출판사 발행/
정가 7,500원/418면/4·6배판



통신분야 책자중에 기초부분부터 실무까지 기술한 책자가 없는 것이 여러 사람들의 불만사항이었다. 하지만 이 책의 발간으로 어느정도 해소 될것으로 본다.

이 책은 주로 데이터 통신이론의 기초와 통신 프로토콜, 그리고 통신 프로토콜의 실제 응용에 대한 전반을 다루고 있다.

전체 4편으로 구성되어 있으며 1편은 데이터통신의 정의, 시스템의 구성요소, 통신소프트웨어 등 데이터통신의 기초부분을 주로 설명하고 있다.

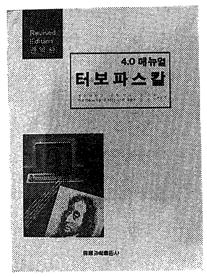
2편은 OSI참조모델, 물리계층, 데이터 링크계층등 각부분을 계층별로 설명하고 있다.

3편에서는 OSI의 구현기술과 동향을 설명하고 있고 미국, 일본, 유럽 등의 추진 동향을 설명한 것이 이채롭다.

4편은 OSI프로토콜의 적용사례를 나열하였다.

터보파스칼 4.0 매뉴얼

강준길·강후길 공역/
홍능과학출판사 발행/
정가 8,500원/628면/4·6배판



이 책은 볼랜드에서 1988년 개정발간한 "TURBO PASCAL 4.0 Owner's HandBook"을 번역한 것으로 사용자가 터보파스칼 프로그램을 작성하고, 컴파일하는 것을 보조하면서, 버전 4.0의 새로운 여러 특징 및 이론을 어떻게 사용하는가를 설명하였다.

이러한 관점에서 쓰여진 이 책은 파스칼을 처음 입문하려는 사람에게 좋은 안내서가 될 것이다.

이 책의 구성은 3부분으로 이루어졌는데, 첫번째는 사용안내로 12장으로 되어 있고, 유니트 및 디버깅과 같은 특별한 특징에 중점을 두어 사용자가 어떻게 책을 이용할 것인가를 설명하고 있다.

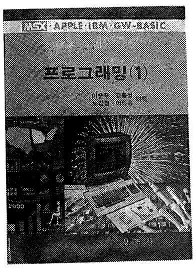
두번째는 터보파스칼에서 사용되는 명령어에 대해 자세히 설명하고 있다. 설명방법은 명령

어를 이용한 짤막한 예제프로그램을 도입하여 명령어를 쉽게 이해할 수 있도록 하였다.

부록부분은 버전3.0과 4.0의 차이, 터보파스칼 4.0과 ANSI파스칼의 비교 등을 설명하고 있다. 또한 미숙한 사용자를 위한 용어와 에러메시지 및 코드도 설명하고 있다.

BASIC 프로그래밍(1)

이순우외 3인 엮음/상조사 발행/
정가 3,500원/328면/4·6배판



컴퓨터 교육과 마인드확산을 위해 초보자들이 부담없이 베이직을 숙지하도록 구성된 책이다.

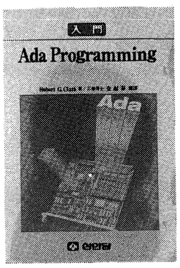
이책은 언어위주의 설명으로 처음 입문하는 사람이 로직개념을 익히기 힘든 다른 책들과는 달리 로직개념도 익힐 수 있도록 배려하고 있다.

제1장에서는 경직되기 쉬운 베이직언어 설명에 앞서 컴퓨터를 재미있게 소개하고 있으며, 다음장은 베이직언어의 특징을 간단히 설명하고 있다. 나머지 부분은 베이직명령을 익히고 연습하여 프로그래밍을 구사할 수 있도록 다루고 있다.

전체의 내용은 순서도와 프로그램리스트를 같이 나열해 순서도와 명령어를 비교하면서 학습할 수 있게 하였다.

Ada Programming

김기태 편역/성안당 발행/
정가 6,000원/326면/국판



신뢰성이 높은 프로그램을 짠다는 것은 쉬운일은 아니다. 하지만 프로그램이 신뢰성이 없다면 프로그램은 무용지물이 되고 만다.

이러한 사항을 고려하여 프로그램을 시작하기 전에 여러가지 문제에 대한 해결방법을 설계해야 한다.

Ada가 가지고 있는 패키지 개념은 신뢰성높은 프로그램을 개발하는 중요한 툴이다. 하지만 다른 언어는 이런 기능이 없다.

이 책은 신뢰성이 높고 이해하기 쉬우며, 또 상당히 효율적인 컴퓨터 프로그램을 설계·작성하는 방법을 설명하고, Ada프로그래밍 언어의 초보적인 설명을 하고 있다.

처음 Ada언어를 배우려는 사람에게 효율적인 프로그래밍 방법을 제공할 것이다.

데이터베이스 환경의 실현 및 관리

우치수 편역/교학사 발행/
정가 10,000원/715면/4·6배판



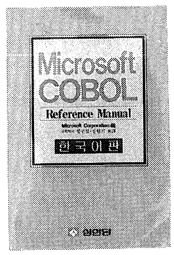
데이터베이스의 기술은 정보처리분야의 앞날에 가장 기초가 된다고 볼 수 있다. 앞으로의 모든 경영·관리는 컴퓨터에 대한 의존도가 더욱 높아질 전망이다. 이에따라 특히 데이터베이스 기술의 이용이 절실히 요구될 것이므로 데이터베이스는 필수 불가결의 요소로 작용될 것이다.

이 책은 James Martin의 'Managing The Data-Base Environment'를 번역한 것으로 실무분야에 주안점을 두고 있다. 특히 소프트웨어의 공학적인 측면을 강조하여 각종 다이어그래밍 기술과 소프트웨어 공학의 배경 및 그 실현을 위한 모델링을 실제 이용자의 입장에서 데이터베이스 시스템의 설계와 관리를 중점적으로 다루고 있다.

따라서 이 책은 데이터베이스를 알고자 하는 학생이나 일반인 그리고 전문기술 요원에 이르기까지 많은 컴퓨터 이용자들에게 좋은 정보를 제공할 것이다.

Microsoft Cobol Reference Manual

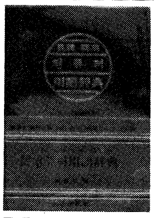
정구철·성원기 공역/성안당 발행/
정가 6,000원/310면/국판



이 책은 마이크로 소프트코볼의 레퍼런스 매뉴얼이다. 1장부터 3장까지는 마이크로소프트 코볼의 소개와 정의, 개념 등을 다루고 7장까지는 코볼에서만 볼 수 있는 4개의 디비젼(Division) 에 대해서 설명하고 있다. 또한 이 부분에서는 일반적으로 사용되는 문(Statement)과 절(Clause) 등의 정의와 사용방법 등을 다루고 있으며 디비젼의 구조를 알파벳순으로도 소개하고 있다.

그리고 프로그램간의 커뮤니케이션, 테이블처리, USE문, 분할, 파일조직, 분류/병합, 복사 그리고 파일과 레코드의 보관 등도 소개하고 있다. PC

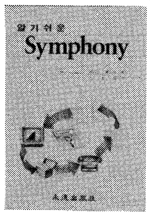
컴퓨터 도서를 전문으로
항상 연구하며 노력하는 자세로



한영 영한 컴퓨터 용어사전

편집부 편 4×6판 값 4,500원

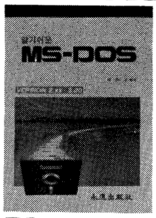
전산, 데이터통신은 물론, 특히 마이크로 컴퓨터 관계의 최신 용어에 역점을 두었으며 권말에 한영색인을 수록하여 사용자 가이드 등 컴퓨터 관계의 원서를 해독하는데 도움이 되도록 엮은 동시에 부록에는 각종 마이컴의 명령 세트 등의 귀중한 연구자료를 수록하여 학생, 실무진 및 연구진을 위한 각종 참고서겸 용어사전이다.



알기쉬운 Symphony

에디슨 웨슬리 저 김 용성 역 4×6배판 값 7,000원

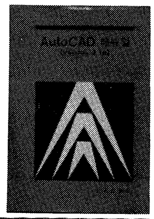
Symphony는 electronic spreadsheet, word processor, database, graph 및 data 통신의 5개 software를 하나의 package로서 실무용 종합 program이다. 즉, Symphony를 사용해서 spreadsheet를 작성하여 data를 입력 조작하고, 문서를 작성 편집, 서식화(format)하고 원격 computer와 교신을 할 수도 있다.



MS-DOS (Version 2.xx～3.20)

이 준구 편저 4×6배판 값 5,000원

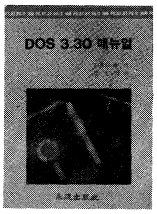
IBM PC XT, AT를 사용하기 위한 기본 도서로 문법에 연연한 설명을 피하고 초보자들도 흥미롭게 MS-DOS를 배울 수 있도록 실습위주로 설명되어 학교나 학원, 회사 등의 교육 교재로 가장 적합하다.



AutoCAD 매뉴얼 (Version 2.18)

Autodesk사 저 국배판 값 12,000원

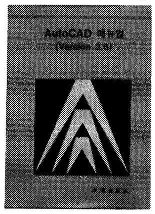
AutoCAD drafting package는 IBM PC 및 그 호환기종에 사용되는 10조 : 1이라는 놀라운 zoom ratio를 갖는 강력한 설계보조용 software로서 우리가 상상할 수 있는 거의 모든 drawing을 신속 정확하게 그려내며, 건축설계, 기기설계, 전자회로설계 등은 물론, freehand sketch에 의한 지도작성, 미술응용, 수학함수의 도표작성 등 광범위한 제도기능을 갖는다.



IBM PC DOS 3.30 매뉴얼

IBM사 저 김 용성 역 4×6배판 값 8,000원

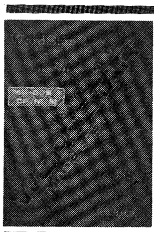
이 책은 본래 DOS 3.30 매뉴얼 원서를 바탕으로 하여 역편된 것이지만, 구 버전 디스켓의 사용자를 위해 버전 3.20 관계 기능도 함께 해설하였으며, 크게 Ⅰ편 DOS User's Guide와 Ⅱ편 DOS Reference로 나뉜다. 특히 프로그래머들이 실무에 있어서 수시로 명령이나 기능에 대한 설명을 쉽게 찾아볼 수 있도록 완전 사전형식으로 엮어진 것이 특징이다.



AutoCAD 매뉴얼 (Version 2.6)

Autodesk사 저 국배판 값 15,000원

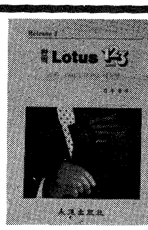
이 책은 IBM PC 및 그 호환기종에 사용되는 10조 : 1이라는 놀라운 Zoom ratio를 갖는 최강의 설계 보조용 S/W인 AutoCAD 버전 2.6을 해설한 기본 매뉴얼로서 구 버전(버전 2.1)에 50여종의 새로운 명령이 추가되었고, 또 기존 명령도 대다수가 크게 보완되었으며, AutoCAD 개정 증보분을 실었다.



WordStar 실무 매뉴얼

Walter A. Ettlin 저 4×6배판 값 4,500원

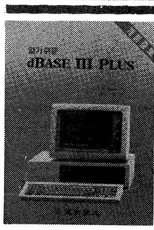
WordStar는 MicroPro International사가 발표한 융통성이 많은 Word Processing Program으로 이 매뉴얼은 일반적으로 사용되는 거의 모든 type의 word processing 기법을 제공해주고 있으며, 특히 이 책에 주어진 연습문제를 완성시킴으로써 얻어지는 기법은 어떤 업무처리용 Word processor에도 활용할 수 있다.



완벽 Lotus 1-2-3 (Release 2)

김 우용 저 4×6배판 값 5,000원

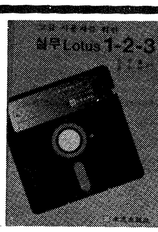
전세계적으로 가장 인기있는 마이크로컴퓨터용 프로그램중의 하나인 Lotus 1-2-3의 가장 최신판인 Release 2를 모델로 초보자도 이해하기 쉽도록 한글과 함께 사용하는 방법을 예를 들어 설명하였다. 1-2-3를 한차원 높게 사용하려는 사람들을 위해서는 내장함수의 사용방법, 매크로 사용법 등을 예를 들어 상세히 설명하고 있다.



알기쉬운 dBASE III PLUS

Robert A. Byers 저 4×6배판 값 6,000원

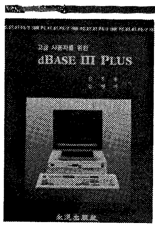
IBM PC를 중심으로 기술된 Database 입문 과정으로 종래의 dBASE Ⅱ와는 달리 메뉴 방식으로 되어 있어서 명령문 작성의 문법을 암기할 필요가 없이 화면에 나타난 항목 중에서 선택하여 쉽게 사용할 수 있으므로 초보자도 이해할 수 있도록 서술되어 입문과정 교재로 적합하다.



실무 Lotus 1-2-3

김 우용 · 홍 수현 편저 260×380m/m 값 10,000원

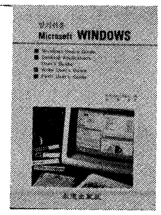
이 책은 사원들의 급여 계산이나 견적서 작성 등 실무에서 활용할 수 있는 프로그램에서부터 경영자 계층에서 경영상태를 분석하고 시뮬레이션할 때의 프로그램에 이르기까지, 지금까지의 방법보다 한차원 높은 단계의 1-2-3 활용 방법을 갈구하던 사용자 계층의 욕구를 만족시켜줄 것이다. 즉시 업무에 활용할 수 있도록 software를 제작 공급하고 있다. (S/W 값 20,000원)



실무 dBASE III PLUS

김 우용 · 김 영미 편저 4×6배판 값 6,000원

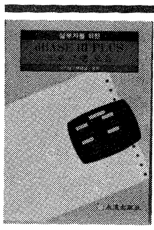
이 책은 dBASE Ⅲ PLUS에 대한 기본 지식을 갖춘 사용자가 실무에 쓸 수 있는 프로그램을 직접 작성할 수 있도록 하였으며, 어떤 업무에서나 쓰기 편리한 「기사정보정리」 및 즉시 사용이 가능한 「재무회계」 프로그램을 자세한 설명으로 수록하였다. 특히 Software를 별도로 제작하여 저렴하게 공급하고 있다. (S/W 값 10,000원)



Microsoft WINDOWS

마이크로 소프트사 저 4×6배판 값 7,000원

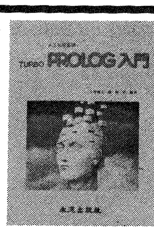
복수의 프로그램을 동시에 사용할 수 있게 해주므로써 마이크로 컴퓨터의 사용자들에게 확장된 DOS라는 평판을 받고있는 WINDOWS 프로그램의 사용방법을 초보자에서부터 한단계씩 단계적으로 학습할 수 있도록 꾸몄다. 한 프로그램에서 다른 프로그램으로 옮겨가는 방법, 데이타를 이송하는 방법 등을 키보드나 마우스를 사용하는 방법과 함께 상세히 설명하고 있다.



dBASE III PLUS 프로그램 모음

이 기성 · 탁 연상 공저 4×6배판 값 9,000원

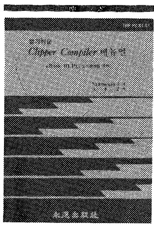
이 책은 한글로 만든 데이타베이스 프로그램(한글 소쿠리), 고객 관리 프로그램, 색인 관리 프로그램, 전화 번호 관리 프로그램, dBASE화일 깨진것 살리기, 문서(저널) 관리 프로그램, 급여 관리 프로그램, 환자(치과) 관리 프로그램, 해운 관리 프로그램 등을 수록하고 별도로 software를 제작하여 공급하므로 업무에 활용할 수 있게 하였다. (S/W 값 20,000원)



인공지능언어 PROLOG 入門

조 동섭 편저 4×6배판 값 5,000원

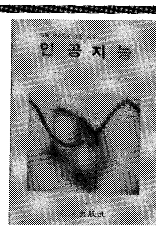
가장 강력한 인공지능언어로 최근 크게 각광을 받고있는 Prolog 언어의 사용법이 단계적 학습과정을 통해 익힐 수 있도록 꾸며졌다. 컴퓨터 사용자에게 슈퍼컴퓨터와 같은 능력을 갖게 해주는 제5세대 언어인 Turbo PROLOG를 이 책을 학습한 뒤에 스스로 Expert System을 디자인할 수 있는 정도로 숙달될 수 있게 될 것이다.



Clipper Compiler 매뉴얼

Nantucket사 저 김 용성 역 4×6배판 값 5,000원

Clipper compiler로 작성된 program은 dBASE와 같은 interpreter로 작성된 program처럼 각 line마다 error check를 하고 기계어로 변환시킴이 없이 모든 program error를 한꺼번에 식별하고 수정할 수 있어 실행속도가 월등하게 빠른 것이 특징이며, 또한 dBASE Ⅲ PLUS의 command와 함수를 사용할 수 있다.

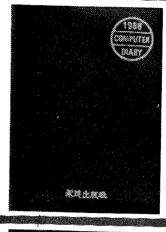


GW-BASIC으로 배우는 인공지능

조 동섭 편저 국판 고급양장 값 4,000원

이 책은 문제해결, 창조성, 초급영어에 의한 컴퓨터와의 대화 등 인공지능을 아는 데 있어서 매우 현실적인 실천에 입각해서 기본이 되는 분야를 소개한 해설서이다.
모든 프로그램은 GW-BASIC에 의해 작성되어 IBM PC/XT/AT 호환기종에서 실행이 가능하도록 하였다.

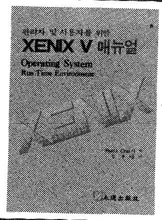
출판하는 저희 「영진출판사」는 컴퓨터 문화창조에 이바지하겠읍니다.



'89 COMPUTER DIARY

편집부 편 4×6배판 고급비닐카바양장 값 3,500원

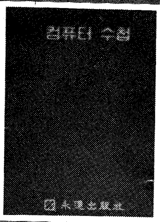
365일 일별메모와 80여 페이지의 메모란, IBM-PC 호환기종을 위한 참고서(MS-DOS, WordStar, Lotus 1-2-3, dBASE Ⅲ Plus, 베이직 명령어 요약표, 한글 DOS 및 한글 카드에서 Graphic Character Cord표), 전국 우편번호, 전국 자동전화번호, 전화번호 메모란, 수도권 지하철 안내도를 수록한 책으로 대량 제작 납품도 가능하다.



XENIX V 매뉴얼

Santa Cruz사 저 김 우용 역 4×6배판 값 7,000원

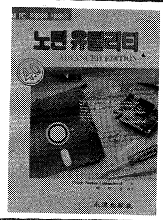
이 책은 SCO의 XENIX V 2.1.3의 설치에서부터 시스템의 증설 및 변경, 실제 사용시에 부딪치는 문제들에 대한 해결 방법 등, 초기 도입에서부터 실용에 이르기까지 시스템 관리자 및 사용자가 알아두어야 할 사항을 설명하는 기본 매뉴얼이다. 다음으로 Operating System 중 User's Guide Guide와 Reference도 곧 출간될 예정이다.



컴퓨터 수첩 (IBM PC용 APPLE, MSX용)

편집부 편 90×150m/m 비닐카바양장 값 1,200원

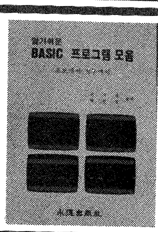
365일 일별메모와 30여 페이지의 메모란, IBM-PC 호환기종을 위한 참고서(MS-DOS, WordStar, Lotus 1-2-3, dBASE Ⅲ PLUS 명령어 및 사용법), 각 기종별 BASIC언어해설, 컴퓨터관련 시험출제기준, 전국 자동전화번호, 전화번호 메모란을 수록한 수첩으로 컴퓨터 학원이나 컴퓨터 메이커용으로 제작 대량 납품도 가능하다.



노턴 유틸리티 4.0 ADVANCED EDITION

탁 연상 편역 4×6배판 값 3,500원

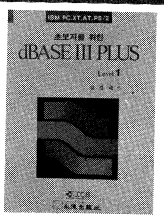
노턴 유틸리티 4.0 ADVANCED EDITION은 지웠던 화일을 다시 복구해 주고, 지웠던 하드 디스크를 다시 복구해 주고, 디렉토리의 화일들의 순서를 마음대로 바꿀 수 있고, 디스크를 검사하고 안전처리해주며, 디렉토리의 구조를 트리구조로 보여주는 등의 다양한 기능을 제공하는 유틸리티 24개로 구성되어 있다.



알기쉬운 BASIC 프로그램 모음

이 기성·탁 연상 공저 4×6배판 값 6,500원

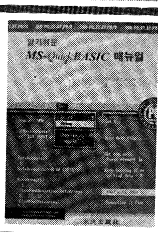
이 책은 개인용 컴퓨터의 사용자들이 가장 쉽게 배울 수 있고, 또 베이직 언어로 유용한 실무용 프로그램을 작성할 수 있도록 각종 실무용 예제를 수록한 베이직 프로그램집이다. 특히 사용자들의 요구에 따라 학습 및 실무에 직접 응용할 수 있도록 software를 제작 보급하고 있다.



초보자를 위한 dBASE III PLUS [1]

김 경태 저 4×6배판 값 6,000원

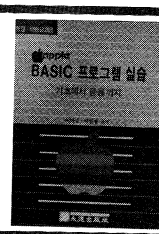
이 책은 dBASE Ⅲ PLUS를 처음 시작하는 분에게 적합하도록 가장 기본적인 사항을 체계적으로 전개하였으며. 과정의 연속성을 유지했으므로 꼭 처음부터 끝까지 읽으면서 dBASE Ⅲ PLUS와 함께 진행하기를 권한다. 곧 Level 2, 3도 출간될 예정이다.



MS-Quick BASIC 매뉴얼

마이크로 소프트사 저 4×6배판 값 7,000원

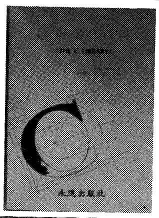
이 책은 마이크로소프트사의 Quick BASIC Version 2.0과 3.0 매뉴얼의 번역서로서, 특히 예제의 이해도를 높이기 위해 한글로 주석을 달았다. Quick BASIC은 기존의 인터프리터 베이직과 거의 완벽한 호환성을 유지하면서도 빠른 속도와 편리한 프로그래밍 환경을 제공하는 강력한 베이직 언어이다.



APPLE BASIC 프로그램 실습

이 기성·탁 연상 공저 4×6배판 값 6,000원

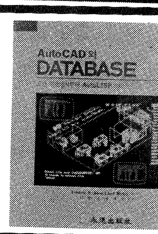
애플 베이직을 사용하여 실제로 프로그램을 작성하는 방법과 실무 프로그램을 보기로 들어서 각 루틴을 자세히 설명하여 왜 이렇게 프로그램을 작성하였는가를 독자가 한 눈에 알 수 있도록 집필하였으며, 현재 실무에서 사용되고 있는 프로그램들이므로 프로그램의 원리를 공부도 하고 실제로 사용할 수도 있어서 일거양득이다.



C 라이브러리

Kris Jamsa저 조 동섭 역 4×6배판 값 5,000원

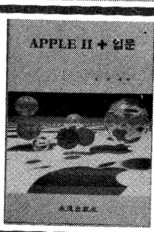
이 책은 C언어로 쓰인 Subroutine이나 함수를 풍부하게 다루고 있다. 이들 Subroutine이나 함수는 모두 범용성이 높기 때문에 어느 분야의 응용 프로그램이나 시스템 프로그램을 작성하더라도 유용하게 사용될 수 있다. 따라서 C프로그래머에게는 필수적인 참고서가 될 수 있을 것이다.



AutoCAD와 DATABASE

김 용성·서 재철 공역 4×6배판 값 6,000원

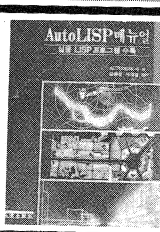
이 책에서는 AutoLISP를 통해서 비 graphic data를 관리하고 조작하는 방법과 DXF file을 조작하여 drawing database를 dBASE나 Lotus 1-2-3와 같은 외부의 database에 link하는 방법을 중심적으로 해설하였다.



APPLE II+입문

송 영태 저 4×6배판 값 3,500원

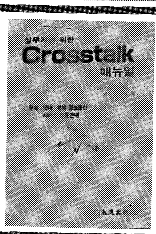
처음 APPLE Ⅱ PLUS를 대하는 초보자가 기본 매뉴얼에서 얻지 못하는 기본 지식에서부터, 주변기기 사용법에 이르기까지 쉽게 설명하였으며, 기본 시스템의 이해와 BASIC 사용에 대한 기본설명과 응용방법, 카세트 레코더와 디스크 드라이브 사용법 APPLE Ⅱ PLUS를 한차원 높여주는 CP/M 사용법에 대하여 명령어의 요약해설과 함께 설명하였다.



AutoLISP 매뉴얼

김 용성·서 재철 편역 국배판 값 7,000원

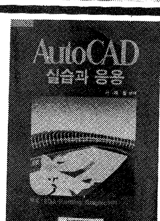
이 책의 제3장에서는 AutoCAD command를 실제로 개발하게 되고, 제4장에서 제6장까지에는 기본이 되는 모든 AutoLISP 함수를 예문을 곁들여 해설하였으며, 제7장에는 AutoLISP 함수를 작성하고 debug할 때 자주 보게 되는 error message를 수록하였다. 부록에는 유용한 많은 실용 program을 수록하여 AutoLISP programming에 있어서 참고가 되도록 하였다.



실무자를 위한 Crosstalk 매뉴얼

Reter W. Gofton저 김용성역 4×6배판 값 5,500원

이 책은 미국 통신 software의 베스트셀러로서 알려진 Crosstalk XVI의 이용법을 알기쉽게 해설한 것으로, 초심자에서 상급자에 이르기까지 각각의 레벨에서 computer 통신을 실행할 수 있도록 체계적으로 엮어져 있으며, 이 책에 소개하는 command file이나 script file을 이용함으로써 통신비용을 크게 절약할 수 있다.



AutoCAD 실습과 응용

서 재철 편저 국배판 값 10,000원

이 책은 실무에서 AutoCAD를 사용하던 사람이 직접 쓴 책으로 Manual식의 피상적인 접근이 아닌 실전 실습과 응용으로서 높은 부가가치를 창출할 수 있는 AutoCAD의 지침서이다. 특히 EGA PAINT라는 좋은 Graphic Program까지 실어서 AutoCAD의 미비점을 보강하여 Graphic 화면을 최대한 살릴 수 있게 하였다.

신간안내

기초부터 응용사례까지 완전해설
BAR CODE시스템 핸드북
박선우 편저 4×6배판 값 5,000원

해설 **PC 통신 입문 I**
정영일 편저 4×6배판 값 5,000원

프로그래머를 위한
dBASE III PLUS (Level 2)
김경태 저 4×6배판 값 6,000원

전자출판
신문과 출판에서 컴퓨터의 이용
이기성 저 4×6배판 값 8,000원

IBM PC 入門
초보자를 위한 가이드
탁 연상 저 4×6배판 값 4,000원

고급 사용자를 위한
실무 Lotus 1-2-3 (제2집)
홍 수현 편저 8절판 값 12,000원

최첨단 도서 전문 출판
永進出版社
☎ 702-1119(代), 719-1119, 719-1190

뭐든지 물어보세요

바이러스란 무엇인지

최근 각종 컴퓨터잡지에는 바이러스에 대한 글들이 소개되고 있습니다. 바이러스란 무엇인지 알려주십시요.

김영준 / 서울 양천구 신월 2동 611-2　성승주 / 인천시 중구 북성동 1가 98

바이러스는 데이터를 파괴시키는 역할을 합니다.

컴퓨터 바이러스란 사용자의 의사에 관계없이 컴퓨터 속의 기억장치로 들어가 악영향을 주는 프로그램을 말합니다.

바이러스란 이름에서도 알 수 있듯이 이들 프로그램들은 자동적으로 다른 컴퓨터에 자기 자신을 복제시키며 순식간에 퍼지는 특징을 갖고 있습니다. 지난해 우리나라에서도 (C) Brain이라는 바이러스 프로그램이 등장해 한동안 소동을 벌인 사실을 독자들도 잘 알고 있으리라 봅니다.

(C) Brain의 경우는 다행히 백신 프로그램이 개발되었지만 더 큰 파괴력을 지닌 프로그램들이 전세계를 통해 많이 나돌고 있기 때문에 위험성은 항상 존재하고 있습니다.

한편 컴퓨터에서는 이들 바이러스 프로그램의 탄생 배경이 실수에 의한 것보다 고의적인 면이 대부분임을 인식하고 범죄문제로까지 취급되고 있으며 이를 대항하기 위한 백신 프로그램 개발에도 많은 관심을 갖고 있습니다.

DBASEⅢ+에서 다른 에디터를 사용하려면

DBASEⅢ+에서 제공되는 에디터인 Modify Command를 이용하여 프로그램을 작성하는데 이용도중에 이상한 에러메세지가 출력됩니다. 즉 프로그램의 스텝수가 많아지니까 "Far to large"라는 에러메세지가 발생됩니다. 다른 에디터를 사용할 수 없는지요. 또 사용할 수 있다면 어떻게 사용하여야 합니까?

성상기/서울시 구로구 시흥3동 976-6

PEⅡ 에디터를 사용하시면 됩니다.

DBASEⅢ+에서는 Modify Command란 자체적인 에디터가 지원되고 있습니다. 그런데 이 에디터는 5000바이트를 넘으면 "Far to large"라는 에러메세지가 나타날 뿐 아니라 에디터로서의 기능이 부족하여 일반적으로 에디터를 사용하려면 DBASEⅢ+에서 제공된 에디터를 사용하지 않고, PEⅡ란 에디터를 많이 사용합니다. 이 PEⅡ 에디터는 구하기 쉽고 기능도 좋아 즐겨 이용하고 있습니다.

하지만 PEⅡ 에디터를 DBASEⅢ+영역속에서 사용하지 않고 별개로 DOS상태에서 에디터를 이용하여 프로그램을 작성하고 다시 DBASEⅢ+를 실행시켜 도트프롬프트 상태에서 사용하고 있는 사용자도 많이 있습니다. 이렇게 사용하면 약간은 불편합니다.

DBASEⅢ+에서는 CONFIG. DB라는 화일을 이용하여 다른 에디터를 사용할 수 있도록 하였읍니다.

PEⅡ 에디터를 CONFIG. DB에 세팅시키려면 다음과 같이 합니다.

우선 CONFIG. DB화일을 Copy명령을 이용하여 작성하여 다음과 같이 입력합니다.

```
C:>COPY CON CONFIG.DB
TEDIT=PE
COMMAND=ASSIST
```

이 이후부터는 DBASEⅢ+에서 PEⅡ 에디터를 사용하실 수 있읍니다.

사용방법은 DBASEⅢ+의 도트프롬프트 상태에서 Modify Command라고 에디터를 호출하면 PEⅡ 에디터를 사용할 수 있읍니다.

VDISK의 설치방법은

저는 IBM-PC/AT를 사용하고 있는데 VDISK를 설치해 보았지만 번번히 설치되지 않았읍니다. 64kB이상 설치하면 제대로 동작이 안되고 360kB는 아예 설치가 되지 않습니다.

VDISK의 자세한 설치방법을 알고 싶습니다.

최종두/충북 청주시 우암동 386-1

확장메모리를 사용할때는 옵션 /E를 사용해야 합니다.

VDISK는 주기억장치의 일부를 떼어서 마치 내부에 드라이브를 하나 더 설치한 것처럼 자유롭게 사용할 수 있읍니다.

VDISK의 설치방법은 DOS의 CONFIG.SYS에 VDISK.SYS를 셋팅하여 설치합니다. VDISK의 설치는 다음과 같이 합니다.

```
C:>COPY CON CONFIG.SYS
DEVICE=VDISK.SYS BUFFER SIZE=bbb
SECTOR SIZE=sss DIRECTORY ENTRIES
=ddd
^Z
C:>
```

여기서 bbb는 가상 디스크사이즈로서 10진 정수의 값으로 지정됩니다. 그리고 디폴트 값은 64kB이고, 가용메모리용량이 64kB 이하이면 에러메세지가 디스플레이됩니다.

sss는 섹터사이즈로 허용되는 사이즈는 128, 256, 512바이트이고, VDISK의 디폴트 값은 128바이트입니다.

ddd는 가상 디스크가 가질 수 있는 디렉토리 엔트리의 수(화일수)로 디폴트 값은 64입니다. 이 값의 범위는 2~512까지 사용이 가능합니다.

/E는 확장메모리를 사용할때 사용되는 옵션으로 확장메모리는 1MB 이상의 메모리를 말합니다. 하나의 VDISK에 대해서 최대 4MB의 확장메모리를 사용할 수 있읍니다. 단 이 옵션은 IBM AT를 사용했을 때만 가능합니다.

그리고 VDISK의 동작이 제대로 되지 않을 때에는 다음 사항을 체크해 보십시요. 첫번째 DOS 버전이 맞아야 하는데 DOS 버전이 제대로 맞는지 체크해 보십시요. 두번째 VDISK.SYS가 깨졌는가를 점검하시고 그 다음에는 CONFIG.SYS의 세팅이 제대로 되었는지 점검해 보십시요.

프로텍트된 프로그램 화일을 풀려면

우선 귀지가 저와 같은 초보자들을 위하여 다양한 기사를 다루어 주시는데 감사드립니다. 다름이 아니라 베이직을 공부하는데 있어 한가지 의문점이 있어 이렇게 펜을 들었읍니다. 드릴 질문은 베이직으로 P옵션 프로텍트를 건 프로그램 화일을 풀려면 어떻게 해야 되는지에 대해서 알고 싶읍니다.

정한신/서울시 도봉구 미아4동 미아아파트 2동 105호

디버거 프로그램을 이용하여 복구합니다.

베이직 프로그램은 세이브 명령시에 P옵션을 두어 프로그램 리스트를 볼 수 없도록 보호하고 있읍니다. 그러나 리스트를 볼 수 있도록 저장해야 하는 경우에도 실수로 P옵션을 붙여 저장시킨다면 큰 낭패가 아닐 수 없읍니다. 이럴 경우 프로텍트를 풀려면 다음과 같이 하면 됩니다.

```
A>debug ↵ ← 디버거 프로그램을 실행시킵니다.
-n   pcqa.lst ← 화일 이름을 지정합니다.
-e   100 fd 66 66 64 04 01 00 00 1a ← 100번지부터 프로텍트를 깨기 위한 프로그램을 입력합니다.
-rcx ← CX레지스터에 저장시킬 프로그램의 바이트수를 기록하기 위해 rcx를 입력한 후 9를 누릅니다.
CX 0000
:9
-W ← 디스켓에 pcqa.lst 화일을 저장합니다.
Writing 00009 bytes
-q ← 디버그 상태에서 도스로 빠져 나옵니다.
```

이렇게 프로텍트가 걸려 있는 프로그램의 리스트를 보기 위한 도구를 만든 다음에 다음과 같은 방법으로 리스트를 보면 됩니다.

```
load " savep.bas" ← 프로텍트된 프로그램을 로드합니다.
OK
list ← 리스트 명령을 내려도 보이지 않습니다.
Illegal function call
OK
bload"pcqa.lst",&h464 ← 앞에서 만들어 놓은 pcpa.lst 화일을 로드합니다.
OK
list ← 그 다음 리스트해보면 됩니다.
```

이렇게 하면 P옵션으로 저장된 프로그램의 리스트를 볼 수 있을 것입니다. 그러나 이 방법은 IBM PC 호환기종중 BIOS부분을 약간 바꿔 놓은 기종에서는 제대로 복구되지 않습니다. 이럴 경우에는 복구가 가능한 기종에서 일단 복구한 후에 원하는 기종에서 실행시켜 보십시요. 원하는 결과를 얻을 수 있을 것입니다.

하드디스크의 최대 용량은

16비트 IBM PC기종에서는 하드디스크의 용량을 어느 정도까지 확장시킬 수 있는지 알려주십시요.

이백진/경기도 광명시 광명6동 374-142 동방연립 12동 101호

736MB까지 가능합니다.

16비트 IBMPC에서는 MS-DOS를 표준 OS로 채택하고 있읍니다. 여기서 OS란 각종 입출력 주변장치를 보다 효율적으로 관리하도록 해주는 운영체제를 말합니다. 그런데 초기의 MS-DOS 버전 1.0에서는 하드디스크를 지원하지 않고 버전 2.0부터 하드디스크를 지원하기 시작하였읍니다. 이때 지원한 용량도 겨우 16M바이트에 불과하였읍니다. 그후 DOS가 계속 개량되어 최근의 MS-DOS 3.3에서는 32MB이상의 하드디스크도 사용이 가능합니다. 단, 대용량의 하드디스크는 전체를 하나로 쓸 수는 없고 여러개의 디스크처럼 논리적으로 분할하여 사용하도록 되어 있읍니다. 이때 기본 도스 분할은 32MB이상을 초과할 수 없으나 그 다음의 확장 도스 분할은 736MB까지의 크기를 가질 수 있도록 되어 있으며, 32MB이하의 논리 디스크로 나누어 사용하여야 합니다.

최초로 등장한 국산캐드 1호

최초의 본격적인 국산 캐드시스템이 한국과학기술원 시스템공학센터에 의해 개발되어 발표되었다. 이번에 발표된 캐드시스템(모델명 : CADMATE)은 시스템공학센터가 과학기술처의 특정연구과제의 일환으로 지난 87년 6월에 개발한 것으로서 당시 일반에게 공개하려 하였으나 이미 많은 외국제품들이 국내시장에 들어와 있었고 국내의 소프트웨어에 대한 인식도가 낮아 불법복제가 성행하던 때라서 발표가 미루어졌다.

그후 시스템공학센터는 과기처와 협의를 거쳐 지난 1월 27일 캐드메이트의 소스코드, 유저매뉴얼, 실행모듈, 프로그래머가이드 등 일체를 일반에게 공개하기로 결정하였다.

18개월씩이나 묶어둔 속사정이야 자세히 알 수는 없지만 아무튼 외제 캐드가 범람하는 현실에 비추어 내용이야 어쩌되었건 순수한 국산 캐드가 등장하였다는 사실은 기쁘기 그지없다. 기능의 장단점은 앞으로 서서히 시간을 두고 사용하다 보면, 들어나겠지만 좋고 나쁨을 젖혀두고 우리손으로 만든 물건이니 많이 이용해야 되지 않을까 한다.

●기능

캐드메이트의 중요한 기능으로는 각종 도면요소의 정의기능(점, 선, 원, 호, 문자열, 곡선, 블록, 스케치 등), 편집기능, 제어기능, 멀티윈도우 등의 기본사양을 갖춰 보다 효율적으로 도면제작을 할 수 있게 되어 있다.

구체적인 시스템구조는 데이터베이스, 도면요소 정의기능의 확충, 변경기능의 확충, 화면제어기능의 확충, 프로그램 기능 및 소프트웨어의 독립성 강화 등을 들 수 있다.

데이터베이스는 화면상의 도면에 관한 정보가 메모리에 4바이트단위의 배열형태로 순차적으로 저장되며, 단위도면요소와 블록도면요소가 동일한 배열상태로 저장된다(그림 1 참조).

여기서 각 도면요소의 기본적인 정보는 헤드부분에 저장되며, 이러한 헤드부분은 모든 도면요소를 항상 기억하고 있다.

또한, 도면요소의 세부적인 기능 및 명령어를 살펴보면 사각형 정의(RECT), PLINE(직선과 호를 연결한 것), SOLID(다각형), TANGENT, FILLET 및 CHAMFER, DIMENSION(선형, 각도, 직경, 반경, 보조기능, 인출선을 이용한 치수기입 등), HATCH(단면요소 발생기능), OSNAP(화면상의 특이점 지정) 등을 갖고 있다.

다음으로 도면요소 변경기능의 확충방법은 LINK, L-EDIT, SPLIT, PMOVE명령을 이용하여 선(Line) 요소의 편집 등을 변경하는데 편리하게 사용할 수 있다.

특히, 멀티윈도우(SETWINDOW 및 WINDOW) 기능의 확충은 화면상의 임의의 위치에 임의의 크기로 영역을 설정하고, 여기에 화면상의 내용을 지시하면, 그 내용이 설정한 윈도우영역에 매핑되어 처리되도록 되어 있다. 또한, 윈도우로 설정한 기존의 내용도 윈도우 명령으로 다시 정의하는 것이 가능하다.

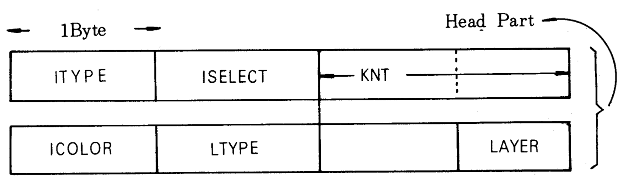
이런 윈도우 기능을 사용한 예는 그림 2에서 보는 바와 같다.

다른 한편으로 캐드메이트는 유사프로그램방식 또는 변수를 이용한 자료입력기능이 개발되어 동종의 다양한 제품(특히 기계부품)의 설계 및 응용에 쓰일 수 있다.

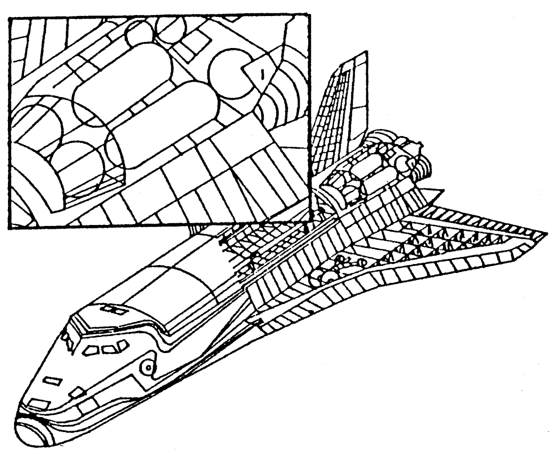
예를 들면, 화일을 작성하여 캐드메이트상에서 배치처리를 하는 경우에 정의된 변수는 그 값이 삽입되며, 정의되지 않은 변수는 수행시 유저가 그 값을 입력하여 사용하도록 되어 있다. 여기서 수행자는 작업에 따라 도면형태를 변경할 수 있는 프로그램기능을 갖고 있다.

프로그램에 있어 변수명은 6자 이내의 문자로 나타내며, 일반 명령어와 구별하기 위해서 변수명앞에 특수문자 "&"를 첨가한다(&문자는 변수명에 포함되지 않음). 제어문으로는 GOTO 라벨, 블록 IF, 논리 IF, FINISH

〔그림 1〕 화면상의 도면내용이 메모리에 배열되는 형태



〔그림 2〕 멀티-윈도우 기능의 예



캐드메이트

등의 명령어가 있으며 과학기술용으로 보편적으로 이용하고 있는 FORTRAN언어와 유사한 것이 쓰이고 있다.

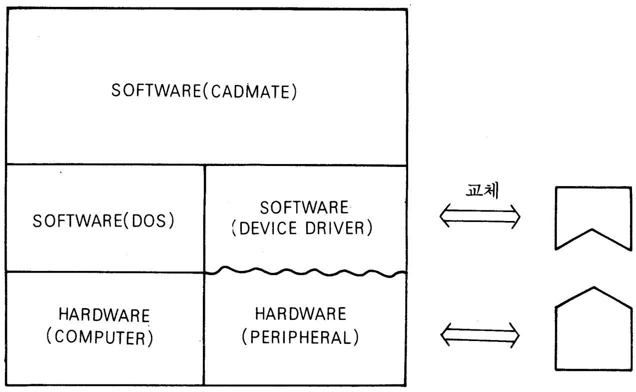
CAD소프트웨어는 그 특성상 다수의 I/O장치를 사용하게 되는데, 중요한 주변기기는 그래픽 전용모니터, 플로터, 디지타이저, 마우스 등이 통용되어 쓰이고 있다. 이들 기기는 다수의 제조업체에서 다양한 특성을 갖는 제품들이 판매되고 있으므로 사용자가 작업실정에 적합한 기기를 자유롭게 선택하여 캐드메이트와 연결하여 사용할 수 있다.

캐드소프트웨어는 실제의 디바이스로부터 독립되게 설계되어 있으므로, 소프트웨어를 용이하게 제작하여야 한다. 이것을 소프트웨어의 독립화라 하며, 기존의 턴키 베이스(Turn Key Base)의 시스템에서는 특정 하드웨어기기만을 지원하며 다른 하드웨어의 이식은 대단히 곤란하게 되어 있다. 하지만, 최근의 경향은 소프트웨어 개발자가 하드웨어로부터의 독립성을 고려하여 개발하고 있다. 물론, 캐드메이트의 개발에도 위에서 언급한 독립성을 갖추고 있다.

즉, 캐드메이트가 가진 독립성으로는 실제의 소프트웨어개발에 가상(Virtual)기기의 개념을 도입하여 이를 실형하는데 있다(그림 3 참조).

그림 3에서 주변기기 하드웨어 교체의 경우를 고려하면, 사용자는 새로운 하드웨어에 해당하는 소프트웨어(디바이스 드라이브로서 캐드메이트와는 별도의 프로그램임)를 설치하여 사용할 수 있다. 여기서 캐드메이트는 그 수행내용에 따라 O/S(여기서는 DOS를 가리킴) 혹은 디바이스 드라이브(가상장치)의 여러 기능을 호출할 수 있다. 이러한 명령들을 디바이스 드라이브가 해석하여 자신이 지원하는 하드웨어의 특성에 맞는 명령어로 전환시켜 실제의 하드웨어를 구동하고 있다.

〔그림 3〕 S/W의 독립성을 위한 설계



확장성

캐드메이트는 2차원 기능을 가지고 있는데 이를 3차원으로 확대할 수도 있다.

또한, 시스템공학센터에서 개발한 엔지니어링 소프트웨어인 KSAP, KISTRAS, K-BUILDING과 연결되어 캐드메이트의 효용을 더욱 확장시킬 수 있다.

KSAP는 구조해석분야에 가장 보편적으로 사용되는 소프트웨어인 SAP(Structural Analysis Program) 시리즈중 SPA V-2를 PC의 특성에 적합하도록 수정, 보완하여 변환한 PC용 프로그램이다. KSAP의 기능은 Blank Common Area가 20000개이고, 3차원 요소의 입력이 가능하며 밴드폭을 최소화하였다. 또한, 캐드메이트를 이용한 그래픽 포스트 프로세싱으로 로드되는 경우에 이를 변형도, 지도제작 등에 효과적으로 사용할 수 있다.

KISTRAS는 2차원용 구조해석 및 설계지원 시스템으로 벽면을 포함한 평면프레임의 설계, 철골 부재설계, 철근 콘크리트 부재설계 등에 쓰인다.

마지막으로 K-BUILDING은 빌딩전용 구조설계 시스템으로서 빌딩 프레임의 입체구조해석과 철골 부재설계 및 철근 콘크리트 부재설계 등을 할 수 있다.

또한, 캐드메이트는 유저가 작성한 프로그램을 해석, 수행하는 인터프리터(Interpreter)가 개발되어 업무특성에 맞도록 최적화를 기할 수 있으며 3차원 설계를 위한 모델의 데이터를 작성하는 별도의 프로그램이 개발되어 투시도, MESH의 발생, 은선제거등의 처리도 할 수 있다.

다른 확장분야로는 그래픽이나, 캐드관련기술을 응용할 수 있는 분야로 3차원 모델의 처리, 솔리드 모델링, 그래픽 에니메이션, 이외에도 각종 설계의 프리포스트 프로세싱분야 등에 쓰일 수 있다.

위에서 언급한 것처럼 캐드메이트는 여러가지 다양한 기능을 갖추고 있다. 앞으로 캐드메이트의 생명은 사용자가 소스프로그램을 분석하여 얼마만큼 각자의 사용목적에 따라 수정, 보완, 발전시켜 사용할 수 있느냐에 달려 있다.

참고로 캐드메이트의 관련자료들 즉, 소스코드, 유저 매뉴얼, 실행모듈, 프로그래머 가이드 등 일체를 한국과학기술원 시스템공학센터(TEL : 965-4842)에서 5만원에 일반에게 판매하고 있다. PC

신제품

50만원대 보급형 컴퓨터
프로 엘리트



가정용, 교육용 및 개인사무용으로 개발된 이 시스템은 OA용인 PRO2000의 개량형으로 IBM-PC / XT와 호환성을 갖고 있다.

이 제품은 첨단 반도체소자로 설계되어 소형 경량화가 실현되었으며 자동화된 대량생산으로 제조원가가 절감되어 경제적인 가격으로 공급가능한 고성능의 보급형 PC이다.

IBM-PC/XT 및 호환기종의 소프트웨어를 그대로 사용할 수 있고 한글/한자 WP, 한글 스프레드시트, 한글데이터베이스 등을 사용할 수 있어 OA업무용에 실용적인 컴퓨터이다.

또한 통신에뮬레이터를 지원함으로써 각종 호스트컴퓨터와 경제적으로 연결이 가능하며 DNS망의 연결을 통해 DACOM의 전자메일 시스템과 생활정보시스템에도 이용할 수가 있다.

그리고 차세대 표준FDD인 720KB 3.5인치 플로피디스크를 내장할 수 있도록 설계하였고 동시에 5.25인치 FDD도 사용이 가능하다.

제품의 규격을 보면 다음과 같다.

- CPU : I8088
- 클럭속도 : 10MHz
- 보조프로세서 : 8087(선택)
- 주기억장치 : ROM : 16KB
  RAM : 256KB
- 보조기억장치 : FDD : 3.5인치(5.25인치 선택)
- 내장기구 : 멀티기능보드(선택)
- 모니터 : 12인치(모노, 칼라)
- 키보드 : 86키

문의처 : 대우통신
전 화 : (02)771-35

네트워크 전용시스템
3STATION/2E



이 제품은 호스트의 방대한 정보를 신속하게 이용할 수 있는 네트워크 전용의 워크스테이션이다.

확장카드나 디스크 드라이브의 필요성을 제거함으로써 전력소모를 최소한으로 줄였고 셋업과 자기진단 프로그램이 롬에 내장되어 있으며 메뉴방식으로 사용할 수 있다는 것이 큰 특징이다.

제품의 규격은 다음과 같다.

- CPU : 80286
- 클럭속도 : 10MHz
- 메모리 : 1MB(최대 5MB)
- 비디오 어댑터 : VGA, EGA, CGA, 허클리스 부착가능
- 시리얼포트 : 2개
- 패러럴포트 : 1개

문의처 : 컴퓨터월드
전 화 : (02)332-0141

휴대용 포켓컴퓨터
PC-E500



이 제품은 Z80 기계어 모니터기능을 내장하여 엔지니어링 분야의 제어 실습용으로 폭넓게 활용이 가능하다.

그리고 외부 제어기기를 제어할 수 있는 Z80 버스 인터페이스를 내장하고 있으므로 각종 입출력 보드를 연결할 수 있는 것이 특징이다.

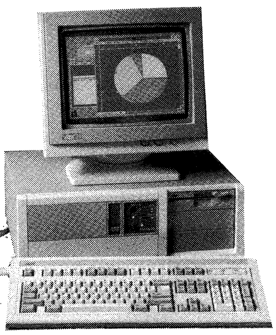
또한 포켓컴퓨터만이 가지고 있는 장점인 건전지 사용, 프로그램 보호기능 그리고 휴대의 간편함 등 시간과 장소에 구애받지 않고 사용할 수있다.

2변수 통계기능을 포함하여 86개의 공학용 함수를 내장하고 있으며 자체에 프로그램 편집이 가능한 24자리 4행의 LCD화면을 가지고 있다. 제품의 규격을 보면 다음과 같다.

- CPU : Z80
- 메모리 : 32KB
- 보조기억장치 : 카세트테이프 레코더 이용 가능
- 화면 : LCD디스플레이(24열×4행)
- 사용언어 : BASIC, 기계어모니터
- 소비전력 : 0.37W
- 크기 : 215×100×18(mm)

문의처 : 샤프전자산업(주)
전 화 : (02)757－6278

인텔 80386SX칩 탑재한
AST 프리미엄 386SX



이 제품은 인텔 386SX칩을 탑재한 퍼스컴으로 기존의 286AT와 386시스템의 장점을 고루 포함한 최신 32비트 제품으로서 두 시스템간의 완벽한 소프트웨어 호환성은 물론 기존 AT가격으로 32비트 시스템기능을 실현할 수 있는 특징을 갖고 있다.

또한 이 프리미엄 워크스테이션 386SX는 모두 6개의 모델(3, 5, 43, 103, 105)로 되어 있으며 다른 제품에 비해 외부 디자인이 매우 콤팩트하고 제로웨이트(Zero Wait) 상태에서 16MHz의 CPU속도를 실현시켰다.

더우기 내부장착 모뎀, 데이터통신 보드, 특수 기능 카드용으로 쓸 수 있는 2개의 8, 16비트 슬롯을 채택하여 6개의 슬롯을 갖는 다른 386SX 퍼스컴과 대등한 기능을 갖도록 설계되어 있다.

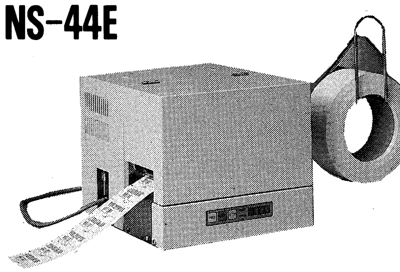
소프트웨어는 MS-DOS 3.2 및 각종 유틸리티 지원이 가능하다.

주요 규격은 다음과 같다.

- CPU : 80386, 20MHz
- 확장슬롯 : 총 7개－32비트 1개, 16비트 4개, 8비트 2개
- 캐쉬 메모리 : 64k바이트 고속(25ns) SRAM
- 메모리 : 표준 1M바이트
- 디스크 드라이브 : 모델에 따라 40M바이트에서 최고 1백10M바이트까지 지원이 가능
- 입출력 포트 : 2 시리얼포트, 1패러럴 포트

문의처 : ICS
전 화 : (02)784－7867

바코드 인쇄용 프린터
NS-44E



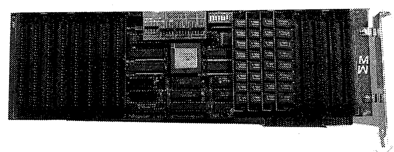
이 제품은 기존 바코드프린터에 비해 인쇄속도가 빠르며 소형, 경량으로 설치가 간편한 것이 특징이다.

또한 기존의 바코드 프린터는 한글이 제대로 처리되지 않는데에 비해 이 제품은 한글 한자의 처리가 자유자재로 가능하다는 것이 큰 특징이다. 제품의 규격은 다음과 같다.

- 인쇄방식 : 열전사 라인도트방식
- 인쇄속도 : 64mm / sec
- 도트밀도 : 7.6 Dots / mm
- 인쇄폭 : 최대 59mm
- 인쇄길이 : 최대 70mm
- 적용 종이폭 : 25mm～70mm
- 적용 종이길이 : 25mm～120mm
- 적용 종이두께 : 0.27mm이하
- 인쇄문자 : 영문자, 숫자, 기호, 한자, 한글(최대2,048자 가능)
- 인쇄크기 : L, M, S
- 인쇄방향 : 0°, 90°, 180°, 270°
- 인터페이스 : RS-232C 시리얼
- 센서 : 투과식 2개 사용

문의처 : 삼미전산(주)
전 화 : (02)583－8523

PC의 계산속도를 높여주는
트랜스퓨터



XT, AT 또는 386에 부착하여 계산속도를 향상시켜주는 새로운 보드이다.

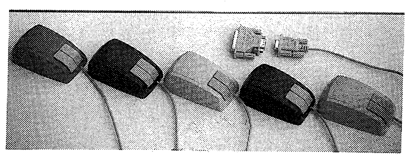
32비트 10MIPS의 계산속도를 가진 T800 RISC 프로세서를 사용하면 최대 40 MIPS까지 계산속도를 증가시킬 수 있으며 코프로세서를 탑재한 AT보다 약15배에서 60배 가량 빠른 처리속도를 갖는다. 또한 MS-DOS상에서 동작되며 메모리도 16MB까지 확장할 수 있다.

제품의 규격은 다음과 같다.

- CPU : T800 RISC 프로세서
- 계산속도 : 최대 40MIPS
- 메모리 : 16MB 확장가능(최대)
- O / S : MS-DOS

문의처 : 다인컴퓨터
전 화 : (02)554－4678

다양한 색상의
컬러 마우스



미국 Vatek사에서 개발한 컬러마우스는 5가지의 다양한 색상중에 사용자가 임의로 선택하여 사용할 수 있다.

이 제품은 정교한 제어기능과 IBM-PC / XT / AT, PS / 2 등 모든 기종에서 사용이 가능한 것이 특징이다.

이 제품의 주요 특징은 다음과 같다.
- 마이크로소프트와 100% 호환 가능.
- PC/XT/AT, PS/2에서 사용가능.
- 사용자임의로 수직, 수평속도 가변가능
- 정교한 제어가능(250DPI)
- 9,25핀 코넥터 양쪽 연결가능(시리얼방식)
- 취향에 따른 색상 선택가능.
- 셋업디스켓으로 임의로 조작가능

문의처 : 한국바테크(주)
전 화 : (02) 566—1108/9

칼라그래픽전용 터미널
DT-9420



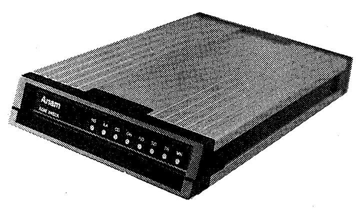
멀티프로세서 아키텍처개념을 사용하여 DEC, Tektronix 4010 / 4014의 그래픽 에뮬레이션 및 VT-340의 문자처리가 가능한 그래픽전용 터미널이다.

제품의 특징을 들면 그래픽의 처리속도를 극대화시킨 점과 서로 다른 호스트로부터 데이터를 받아 동시에 처리할 수 있는 화면동시분할기능(Dual Host Support) 등을 들 수 있다.

제품의 주요 특징은 다음과 같다.
- 그래픽프로세서용CPU : TMS 34010
- 텍스트 프로세서용CPU : i80C31
- 키보드제어CPU용 : i8031
- 해상도 : 800H×600V 픽셀
- 텍스트 디스플레이 메모리 : 최대8페이지
- 리얼타임클럭내장
- 2개의 섹션으로 화면동시 분할
- ReGIS, Sixel, Tek4010 / 4014 에뮬레이션
- VT-340 문자처리 기능
- 4비트플랜 2페이지 지원
- 데이터 전송속도 : 최대38.4kBps.

문의처 : 대우통신
전 화 : (02) 771—35

다기능 모뎀
ADM2450A



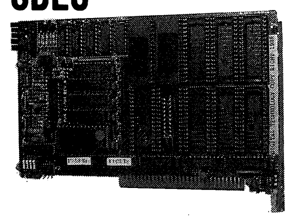
비동기 / 동기 기능 및 다이얼업 / 전용선 기능을 모두 겸비한 2400bps급 모뎀이다.

이 제품은 에러정정기능과 다이얼 압축기능도 가능하며 실제 전송효율을 배가시켜 4800bps의 속도로 데이터전송할 수 있다.

제품의 규격은 다음과 같다.
- 호환성 : CCITT V.22bis, CCITT V.22, CCITT V.21 Bell 212A, Bell 103
- 적용 : 동기, 비동기, 이진시리얼, 자동응답기능
- 변조 : QAM-2400bps
PSK-1200bps
FSK-0~300bps
- 전송레벨 : −1~−16dbm
- 디지탈 인터페이스 : RS-232C(CCITT V.24 / V.28)
- 응용 소프트웨어 : CROSSTALK MK. 4, PC-TALK, SMARTCOM-Ⅱ

문의처 : 아남전기
전 화 : (032) 529—7901

SNA 3270에뮬레이터
DT-SDLC



IBM이나 FACOM의 메인프레임과 퍼스널컴퓨터와의 통신을 구성하는데 있어서 가장 큰 문제점은 IBM-PC는 기본 통신규격을 비동기식으로 하는 반면, 메인프레임은 통신의 효율과 아울러 SDLC / HDLC같은 동기통신방식을 이용하고 있다는 점이다.

SNA 3270에뮬레이터(Emulutor)는 이러한 점 등을 보완하여 PC의 STAND-ALONE 기능을 살리면서 터미널 에뮬레이션도 가능하도록 한다.

이 제품의 기능은 다음과 같다.

–소프트웨어
- SDLC커넬
- 3270 에뮬레이터
- 메인프레임 한글코드 변환프로그램

–하드웨어
- SIO : Z-8530
- 통신방식 : RS—232C(모뎀 인터페이스)
- 최대통신속도 : 19,200bps

문의처 : 디지탈테크놀러지
전 화 : (02) 581—1511

한글의 표현방식

양왕성 /성균관대 마이컴연구회

컴퓨터를 처음 구입하였을 때 초보자라면 누구나 한글의 사용이 가장 궁금할 것 이다. 그러나 여기 저기서 이런방식 저런방식으로 서너가지의 한글 쓰는 방법을 들으면 오히려 한글 쓰는 것이 복잡해 보이고 혼란스럽게 생각하기 쉽다.

여기서 간단하게나마 한글을 사용하는 방법과 사용에 있어서의 문제점을 알아보고 한글을 효과적으로 사용하는데 도움이 되었으면 하는 것이 필자의 의도임을 밝혀둔다. 한글을 쓰는 방법에는 한글카드를 이용한 하드웨어적인 방법과 비교적 기종에 제약을 받지 않는 소프트웨어적인 방법이 있다.

한글카드란, 말 그대로 한글을 쓸 수 있게 해주는 하드웨어적인 기판을 지칭하는것으로 각 컴퓨터 회사마다 특징있는 한글카드를 개발해 놓고 있다. 대표적으로 삼보 한글카드, 옴니 한글카드, 큐닉스 한글카드 등 여러종류가 있으나 여기서는 우선 대부분의 사용자가 세운상가를 통해서 컴퓨터를 구입하고 있다고 생각하여 세운상가 한글카드에서의 한글 사용방법을 설명한 후에 타기종에서의 한글카드의 종류와 특징을 비교해 보고자 한다.

초보자들의 고민거리 중의 하나가 소프트웨어와 하드웨어의 한글을 혼동하는 것이다. 세운상가 한글카드는 허큘리스 그래픽카드에 업혀 있는(붙어 있는) 기이한 모양을 하고 있다. 초기의 한글카드는 허큘리스와 같은 크기로 두개가 붙어 있었는데, 그 크기가 점점 작아져서 요즘은 하나의 칩으로 허큘리스에 장착한 상태의 한글카드(이를 합쳐서 세운한글카드라 한다.)도 나오고 있다.

세운한글카드에서 한글을 쓰는 방법에는 소프트웨어와 하드웨어의 두가지 방법이 있다. 하드웨어적인 방법은 한글카드의 한글을 사용하는 방법이고 소프트웨어적으로 한글을 쓰는 방법에는 여러가지가 있으나 여기서는 대체로 널리 사용되고 있는 삼보컴퓨터의 NKP.COM(보석글 Ⅱ에서 이용)을 이용한 방법을 설명한다. 표 1은 하드웨어와 소프트웨어 한글의 간략한 비교를 나타낸 것이다.

하드웨어에 의한 한글

하드웨어에 의한 한글구현의 방법은 2바이트의 한글코드를 롬에 저장하여 이것을 텍스트화면에 출력시키게 한다.

세운한글카드는 2바이트 7비트 완성형 방법으로 한글을 구현하는데 쓸 수 있는 한글은 모두 1339자로 표 3과 같다. 표 3에서 선택된 가장 많이 사용되는 글자를 롬에 내장하여 텍스트상에 출력하는 방법으로 사용자는 속도를 느끼지 못한다.

한글카드는 영문전용모드와 한글모드를 토글하는 스위치를 가지고 있는데 대부분 컴퓨터 뒤에 스위치가 붙어 있다. 이 스위치를 ON 상태로 놓으면 한글을 표현할 수 있는 상태가 되는데 이것은 단지 한글로 표현된 자료를 볼 수 있을 뿐이며 한글을 입력하려면 hskey를 실행시켜야 한다.

hskey는 램상주 프로그램으로 이 프로그램의 기능은 한글 타이핑을 받아서 이에 맞는 코드조합으로 바꿔 주는 것이다. 예를 들면 한글모드에서 '땅'을 입력했을 때 이것을 'gD'로 바꿔주는 것으로 표 2와 3을 보면 '땅'과 'gD'는 같은 코드이고 단지 한글 스위치의 ON / OFF상태 차이만 있음을 알 수 있다.

그러므로 한글이 필요할 때 한글을 입력할 수 있게 해준다. 이때부터는 실제 영문소프웨어를 실행하다가 한글이 필요하면 〈Alt〉-Z(Alt키와 Z키를 동시에)을 누르면 프롬프트의 모양이 작은 사각형으로 변하고 이때부터는 한글 자판을 보고 한글을 입력하면 된다. 다시 영문이 필요하면 〈Alt〉-Z을 재차 누른다. 즉 〈Alt〉-Z은 토글동작이므로 한글 / 영문 변환이 필요할 때마다 한번씩 눌러주기만 하면 된다.

hskey를 실행하면 허큘리스카드의 프린터포트에 연결된 프린터로 한글이 출력된다. 프린터에 따라 차이가 있으나 엡슨계열에서 hskey를 실행하지 않으면 화면으로는 한글이 나와도 프린터로 한글이 출력되지 않는다. 그

〔표 1〕 하드웨어 한글과 소프트웨어 한글의 비교

	하드웨어 한글	소프트웨어 한글
준비되야할 것	세운허큘리스한글카드 HSKEY. COM(or HGKEY. COM)	HGC. COM(or HGC. EXE) NKP. COM
세팅(활성화)	한글스위치 ON A>hskey	A>hgc full A>nkp
한글입력을 하려면	〈Alt-Z〉 } 토글	〈Alt-F9〉
다시 영문입력을 하려면	〈Alt-Z〉	〈Alt-F10〉
한글모드 제거	한글스위치 OFF, 컴퓨터 리세트	〈Alt-F1〉
사용가능한 소프트웨어	Sidekick, Turbo 시리즈, Lotus123, DBASE 그외 텍스트모드 S/W	GWBASIC G2(보석글)
단점	쓸 수 있는 한글이 제한되어 있음	텍스트모드 보다 느림
주의	hskey를 두번 실행하면 컴퓨터는 다운 됨	hskey를 실행한 상태로 NKP를 실행하면 프린터 출력이 안됨.

〔표 2〕 세운한글카드에서 한글 스위치의 OFF상태

```
_! _" _# _$ _% _& _' _( _) _* _+ _, _- _. _/ _0 _1 _2 _3 _4 _5 _6 _7 _8 _9 _: _; _< _= _> _?
_@ _A _B _C _D _E _F _G _H _I _J _K _L _M _N _O _P _Q _R _S _T _U _V _W _X _Y _Z _[ _₩ _] _^ __
_` _a _b _c _d _e _f _g _h _i _j _k _l _m _n _o _p _q _r _s _t _u _v _w _x _y _z _{ _| _} _~
`! `" `# `$ `% `& `' `( `) `* `+ `, `- `. `/ `0 `1 `2 `3 `4 `5 `6 `7 `8 `9 `: `; `< `= `> `?
`@ `A `B `C `D `E `F `G `H `I `J `K `L `M `N `O `P `Q `R `S `T `U `V `W `X `Y `Z `[ `₩ `] `^ `_
`` `a `b `c `d `e `f `g `h `i `j `k `l `m `n `o `p `q `r `s `t `u `v `w `x `y `z `{ `| `} `~
a@ aA aB aC aD aE aF aG aH aI aJ aK aL aM aN aO aP aQ aR aS aT aU aV aW aX aY aZ a[ a₩ a] a^ a_
b@ bA bB bC bD bE bF bG bH bI bJ bK bL bM bN bO bP bQ bR bS bT bU bV bW bX bY bZ b[ b₩ b] b^ b_
c@ cA cB cC cD cE cF cG cH cI cJ cK cL cM cN cO cP cQ cR cS cT cU cV cW cX cY cZ c[ c₩ c] c^ c_
d@ dA dB dC dD dE dF dG dH dI dJ dK dL dM dN dO dP dQ dR dS dT dU dV dW dX dY dZ d[ d₩ d] d^ d_
e@ eA eB eC eD eE eF eG eH eI eJ eK eL eM eN eO eP eQ eR eS eT eU eV eW eX eY eZ e[ e₩ e] e^ e_
f@ fA fB fC fD fE fF fG fH fI fJ fK fL fM fN fO fP fQ fR fS fT fU fV fW fX fY fZ f[ f₩ f] f^ f_
g@ gA gB gC gD gE gF gG gH gI gJ gK gL gM gN gO gP gQ gR gS gT gU gV gW gX gY gZ g[ g₩ g] g^ g_
h@ hA hB hC hD hE hF hG hH hI hJ hK hL hM hN hO hP hQ hR hS hT hU hV hW hX hY hZ h[ h₩ h] h^ h_
i@ iA iB iC iD iE iF iG iH iI iJ iK iL iM iN iO iP iQ iR iS iT iU iV iW iX iY iZ i[ i₩ i] i^ i_
j@ jA jB jC jD jE jF jG jH jI jJ jK jL jM jN jO jP jQ jR jS jT jU jV jW jX jY jZ j[ j₩ j] j^ j_
k@ kA kB kC kD kE kF kG kH kI kJ kK kL kM kN kO kP kQ kR kS kT kU kV kW kX kY kZ k[ k₩ k] k^ k_
l@ lA lB lC lD lE lF lG lH lI lJ lK lL lM lN lO lP lQ lR lS lT lU lV lW lX lY lZ l[ l₩ l] l^ l_
m@ mA mB mC mD mE mF mG mH mI mJ mK mL mM mN mO mP mQ mR mS mT mU mV mW mX mY mZ m[ m₩ m] m^ m_
n@ nA nB nC nD nE nF nG nH nI nJ nK nL nM nN nO nP nQ nR nS nT nU nV nW nX nY nZ n[ n₩ n] n^ n_
o@ oA oB oC oD oE oF oG oH oI oJ oK oL oM oN oO oP oQ oR oS oT oU oV oW oX oY oZ o[ o₩ o] o^ o_
p@ pA pB pC pD pE pF pG pH pI pJ pK pL pM pN pO pP pQ pR pS pT pU pV pW pX pY pZ p[ p₩ p] p^ p_
q@ qA qB qC qD qE qF qG qH qI qJ qK qL qM qN qO qP qQ qR qS qT qU qV qW qX qY qZ q[ q₩ q] q^ q_
r@ rA rB rC rD rE rF rG rH rI rJ rK rL rM rN rO rP rQ rR rS rT rU rV rW rX rY rZ r[ r₩ r] r^ r_
s@ sA sB sC sD sE sF sG sH sI sJ sK sL sM sN sO sP sQ sR sS sT sU sV sW sX sY sZ s[ s₩ s] s^ s_
t@ tA tB tC tD tE tF tG tH tI tJ tK tL tM tN tO tP tQ tR tS tT tU tV tW tX tY tZ t[ t₩ t] t^ t_
u@ uA uB uC uD uE uF uG uH uI uJ uK uL uM uN uO uP uQ uR uS uT uU uV uW uX uY uZ u[ u₩ u] u^ u_
v@ vA vB vC vD vE vF vG vH vI vJ vK vL vM vN vO vP vQ vR vS vT vU vV vW vX vY vZ v[ v₩ v] v^ v_
w@ wA wB wC wD wE wF wG wH wI wJ wK wL wM wN wO wP wQ wR wS wT wU wV wW wX wY wZ w[ w₩ w] w^ w_
x@ xA xB xC xD xE xF xG xH xI xJ xK xL xM xN xO xP xQ xR xS xT xU xV xW xX xY xZ x[ x₩ x] x^ x_
y@ yA yB yC yD yE yF yG yH yI yJ yK yL yM yN yO yP yQ yR yS yT yU yV yW yX yY yZ y[ y₩ y] y^ y_
z@ zA zB zC zD zE zF zG zH zI zJ zK zL zM zN zO zP zQ zR zS zT zU zV zW zX zY zZ z[ z₩ z] z^ z_
{! {" {# {$ {% {& {' {( {) {* {+ {, {- {. {/ {0 {1 {2 {3 {4 {5 {6 {7 {8 {9 {: {; {< {= {> {?
{@ {A {B {C {D {E {F {G {H {I {J {K {L {M {N {O {P {Q {R {S {T {U {V {W {X {Y {Z {[ {₩ {] {^ {_
{` {a {b {c {d {e {f {g {h {i {j {k {l {m {n {o {p {q {r {s {t {u {v {w {x {y {z {{ {| {} {~
|! |" |# |$ |% |& |' |( |) |* |+ |, |- |. |/ |0 |1 |2 |3 |4 |5 |6 |7 |8 |9 |: |; |< |= |> |?
|@ |A |B |C |D |E |F |G |H |I |J |K |L |M |N |O |P |Q |R |S |T |U |V |W |X |Y |Z |[ |₩ |] |^ |_
|` |a |b |c |d |e |f |g |h |i |j |k |l |m |n |o |p |q |r |s |t |u |v |w |x |y |z |{ || |} |~
}! }" }# }$ }% }& }' }( }) }* }+ }, }- }. }/ }0 }1 }2 }3 }4 }5 }6 }7 }8 }9 }: }; }< }= }> }?
}@ }A }B }C }D }E }F }G }H }I }J }K }L }M }N }O }P }Q }R }S }T }U }V }W }X }Y }Z }[ }₩ }] }^ }_
}` }a }b }c }d }e }f }g }h }i }j }k }l }m }n }o }p }q }r }s }t }u }v }w }x }y }z }{ }| }} }~
~! ~" ~# ~$ ~% ~& ~' ~( ~) ~* ~+ ~, ~- ~. ~/ ~0 ~1 ~2 ~3 ~4 ~5 ~6 ~7 ~8 ~9 ~: ~; ~< ~= ~> ~?
~@ ~A ~B ~C ~D ~E ~F ~G ~H ~I ~J ~K ~L ~M ~N ~O ~P ~Q ~R ~S ~T ~U ~V ~W ~X ~Y ~Z ~[ ~₩ ~] ~^ ~_
~` ~a ~b ~c ~d ~e ~f ~g ~h ~i ~j ~k ~l ~m ~n ~o ~p ~q ~r ~s ~t ~u ~v ~w ~x ~y ~z ~{ ~| ~} ~~
```

리고 hskey를 실행한 후에도 다음과 같이 〈ESC〉 h 〈ESC〉 W를 프린터에 세트해 주어야 한다.

(파스칼에서의 한글 세트)

```
begin
  writeln(lst,chr(27)+'h'+chr(27)+'w'
end
```

(베이직에서의 한글 세트)

```
10 LPRINT CHR$(27)+'h'+CHR$(27)+'w'
```

hskey를 두번 실행시키면 폭주(컴퓨터 동작정지)하므로 주의해야 한다. hskey는 자체적으로 제거가 안되므로 전문 램상주 제거프로그램을 사용하거나 그렇지 않으면 컴퓨터를 리세트 시켜야 한다.

여기서 hskey대신 hgkey를 써도 상관은 없으나 hgkey는 한글/영문 토글을 숫자기판의 '*'을 눌러 하는 것 이외에 사용방법은 hskey와 동일한데 가끔 입력키를 받아들이지 않고 이유없이 폭주하는 등 약간의 문제가 있어 요즘은 대부분 hskey를 쓴다.

필자의 경험에 의하면 hgkey의 에러를 보강한 것으로 거의 에러가 없이 작동하므로 hskey를 권하고 싶다.

한글이 하드웨어적으로 구현되면 한글코드의 범위가 특수코드(!-~)와 영문자를 벗어나지 않으므로, 대부분의 영문소프트웨어를 수정하지 않고 한글을 쓸 수 있다는 장점이 있다. 그러나 텍스트모드에서 한글을 쓰고 있기 때문에 그래픽모드를 쓰는 소프트웨어에서는 한글을 쓸 수 없다는 단점도 있다. 그 예로 로터스 1-2-3에서 한글 데이터 입력과 프린터 출력은 되나, 그래프를 그릴 경우 한글 스위치를 내려 놓은 것처럼 한글이 깨져서(?) 나온다.

또한 한글을 입력시킬 때 가끔 안나오는 한글도 있다. 이 문제는 2바이트 7비트 완성형 한글의 근본적인 문제인데 자세한 사항은 한글표현방식에서 다시 언급하겠다.

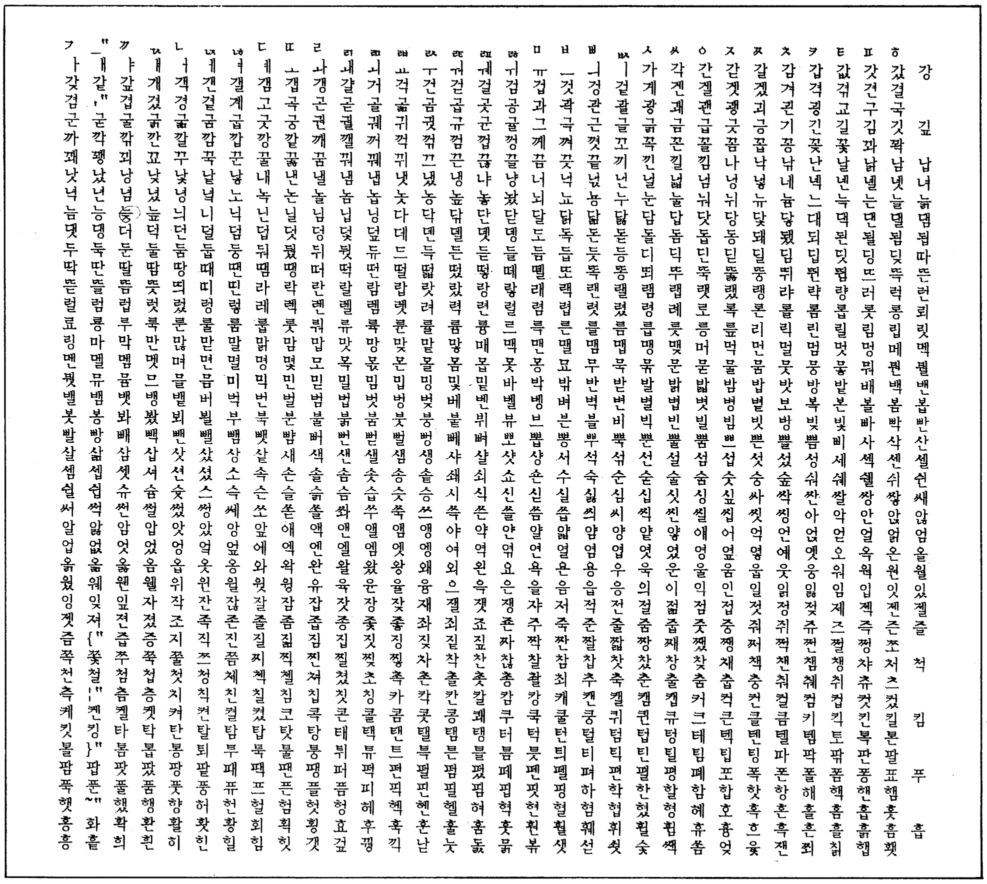
소프트웨어에 의한 한글

소프트웨어에 의한 한글의 구현방법은 그래픽화면 상태에서 한글의 모양을 나타내므로 텍스트상태의 하드웨어에 의한 방법보다 속도면에서 떨어진다. 그러나 2바이트로 한글의 초성, 중성, 종성을 조합하여 표현하기 때문에 모든 한글을 나타낼 수 있다.

NKP를 사용한 소프트웨어적인 방법은 하드웨어적인 제약을 거의 받지 않기 때문에 한글카드가 없는 허큘리스카드를 가진 컴퓨터에서도 작동을 하므로 허큘리스카드를 가진 사용자들에게는 유용할 것이다. 또 워드프로세서로 보석글을 가장 많이 쓰는 현실을 볼 때 이 방법의 한글은 한글 데이터의 호환성을 유지한다는 면에서도 의미가 있다.

NKP는 허큘리스카드의 그래픽모드에서 한글을 사용하므로 허큘리스카드를 그래픽이 가능하도록 세트시켜야 하는데 이 프로그램이 HGC.COM이다. HGC.COM은 한번만 실행시키면 다

〔표 3〕 세운한글카드에서 한글 스위치의 ON상태



시 실행하지 않아도 되는데 그래픽을 쓰는 대부분의 소프트웨어를 실행하기 전에 표 1과 같이 hgc full를 실행한다. (자세한 것은 본지 89년 2월호 '허큘리스카드'란을참조하기 바란다.)

NKP를 실행시키면 화면이 그래픽모드로 바뀌면서 모든 글자를 그래픽으로 처리한다. HGC를 실행하지 않고 NKP를 실행하면 화면이 붕뜨는 것같이 엉망이 되는데 이때는 〈Alt〉-F1을 눌러 NKP를 제거하고 hgc full부터 다시 실행한다.

NKP로 화면이 처리되면 한글이 필요할 때 〈Alt〉-F9키를 누르면 한글 입력모드가 되고 〈Alt〉- F10을 누르면 영문 입력모드가 된다. 한글을 입력할 때 〈Enter〉를 치면 한글 입력모드가 풀리고 다시 영문모드로 돌아오는데 〈Enter〉를 칠 때 한글모드가 풀리지 않게 하려면 NKP를 실행시킬 때 다음과 같이 G 옵션을 준다.(보석글을 쓸 때 이렇게 한다.)

```
C:\>NKP G
```

NKP는 화면을 그래픽상태로 처리하기 때문에 BIOS를 이용하지 않고 화면을 직접 다루는 프로그램에서는 화면위에 점선이 생기면서 원하는 화면이 안나온다. 프로그램이 제대로 동작해도 화면만 안 나오는 경우가 있다. 이때는 프로그램 종료 순서로 키를 눌러 종료한다. 그 예로 NKP 상태에서 pctools을 실행하면 앞에서 설명한 것과 같은 현상이 일어나는데 이때는 pctools의 종료인 〈ESC〉를 누르면 다시 도스상태로 돌아온다.

NKP를 실행시키고 프린터로 한글을 출력하려면 프린터에 한글을 세트한다. 이 방법은 하드웨어에서 설명한 방법과 동일하다. 물론 보석글같은 워드프로세서에서는 따로 세트 프로그램을 실행시키지 않아도 된다.

NKP에서 사용할 수 있는 프로그램은 범용으로 쓰이는 것에 GWBASIC 또는 보석글 등이 있으나 상당히 제한적이다. GWBSIC 에서는 그림과 함께 한글을 쓸 수 있으므로 많이 이용하고 있고, 이를 이용한 소프트웨어도 많다. 터보시리즈(터보 파스칼, 터보 베이직, 터보 C 등)에서는 직접 동작하지 않으나 프로그램을 컴파일해서 NKP상에서 동작할 수는 있다. 예를 들면 소스 프로그램을 보석글로 만들어서 보관하고 터보에서 로드하여 디스크 컴파일 옵션으로 컴파일하고 도스에서 실행할 수 있다는 말이다.

보석글 사용자가 많아서 여타의 다른 워드프로세서에서는 NKP의 한글코드와 호환을 유지하기 위해 대부분 한글코드 변환유틸리티를 포함하고 있다.

NKP상에서 영문 소프트웨어를 사용하다보면 텍스트상태의 영문그래픽 캐릭터(┌, ┼, ║, ╪ …등)가 이상한 한글로 나타난다. (뷁, 똛, 귅…) 그래픽캐릭터는 코드값이 127보다 커서 첫 비트가 1로 되어 있어 이것을 한글로 표현하게 된다(그래픽 코드표 참조). 그러므로 화려한 화면을 위해 그래픽코드를 많이 쓰는 영문 소프트웨어에서는 그래픽코드가 다음 글자와 같이 겹쳐서 한글로 나타나므로 그대로 쓰기에는 문제가 많을 수밖에 없다. 즉 이 방법에서는 그래픽코드를 쓰는 방법이 틀리며 그래픽코드를 쓰기 위해서 그래픽 판별코드를 두고 있다. 그래서 NKP 경우 그래픽캐릭터(캐릭터값 : 212)가 먼저 나오면 다음 코드와 함께 2바이트의 그래픽코드로 나타나는 것이다.

참고로 NKP는 삼보한글카드를 모의 동작하도록 만들어져 있는데 삼보한글카드는 2바이트 8비트 조합형 한글이다. 구버전은 THP이고, 신버전이라 할 수 있는 NKP는 한글 글씨체가 예쁘고 사용하기가 쉬워서 많이 이용되고 있다. 또 사용목적에 따라 NKP는 변형 버전(통신용

의 NKP는 〈Alt〉로 그래픽모드를 입력하지 못한다)도 많이 있다.

한글 표현방식

이제는 어느정도 한글의 사용에 관하여 이해하였을 것이다. 그러면 한글이 컴퓨터상에서 어떻게 표현되는지 알아보자.

원래 영문자만 표현되는 컴퓨터에서 한글 표현이 문제가 되는 것은 IBM사에서 컴퓨터를 설계할 때 한글이 쓰일 것을 고려하지 않은 탓이다. 만약 우리나라에서 컴퓨터가 발명되었다면 아마도 미국에서는 우리가 현재 겪고 있는 것과는 반대의 문제점인 한영 변환 문제가 대두되고 있을 것이다.

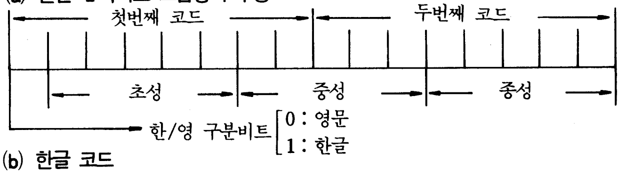
지금까지의 한글카드로 보급되고 있는 한글 표현방식은 대부분 한글 한자를 2바이트로 구성하는 것인데 이것을 다시 완성형과 조합형으로 구분한다. 완성형은 또 7비트 완성형과 8비트 완성형으로 나눈다. 그런데 이들은 모두 정렬(sort)를 염두에 두고 개발되어 있다는 것이 특이하다.

(1) 2바이트 조합형

이 방법은 일반 키보드배열 문자일 경우 8비트 중 첫비트(최상위비트)가 0으로 되어 있는데 이것이 1이면 다음 바이트와 함께 한글로 인식한다. 2바이트 16비트중 첫비트는 한글 구분코드이고 나머지 15비트는 5비트씩 초성, 중성, 종성으로 배당된다. 각 5비트는 $2^5=32$가지의 조합을 만들 수 있으므로 충분히 초성 19가지, 중성 21가지, 종성(받침) 27가지를 수용할 수 있다(그림 1 참조).

〔그림 1〕 2바이트 조합형의 글자표현

(a) 한글 2바이트 조합형의 구성



(b) 한글 코드

Bit 4 3 2 1 0	초 성	중 성	종 성
0 0 0 0 0	Fill Code	Fill Code	Fill Code
0 0 0 0 1	Fill Code	Fill Code	Fill Code
0 0 0 1 0	ㄱ	Fill Code	ㄱ
0 0 0 1 1	ㄲ	ㅏ	ㄲ
0 0 1 0 0	ㄴ	ㅐ	ㄳ
0 0 1 0 1	ㄷ	ㅑ	ㄴ
0 0 1 1 0	ㄸ	ㅒ	ㄵ
0 0 1 1 1	ㄹ	ㅓ	ㄶ
0 1 0 0 0	ㅁ	Fill Code	ㄷ
0 1 0 0 1	ㅂ	Fill Code	ㄹ
0 1 0 1 0	ㅃ	ㅔ	ㄺ
0 1 0 1 1	ㅅ	ㅕ	ㄻ
0 1 1 0 0	ㅆ	ㅖ	ㄼ
0 1 1 0 1	ㅇ	ㅗ	ㄽ
0 1 1 1 0	ㅈ	ㅘ	ㄾ
0 1 1 1 1	ㅉ	ㅙ	ㄿ
1 0 0 0 0	ㅊ	Fill Code	ㅀ
1 0 0 0 1	ㅋ	Fill Code	ㅁ
1 0 0 1 0	ㅌ	ㅚ	Fill Code
1 0 0 1 1	ㅍ	ㅛ	ㅂ
1 0 1 0 0	ㅎ	ㅜ	ㅄ
1 0 1 0 1	Fill Code	ㅝ	ㅅ
1 0 1 1 0	Fill Code	ㅞ	ㅆ
1 0 1 1 1	Fill Code	ㅟ	ㅇ
1 1 0 0 0	Fill Code	Fill Code	ㅈ
1 1 0 0 1	Fill Code	Fill Code	ㅊ
1 1 0 1 0	Fill Code	ㅠ	ㅋ
1 1 0 1 1	Fill Code	ㅡ	ㅌ
1 1 1 0 0	Fill Code	ㅢ	ㅍ
1 1 1 0 1	Fill Code	ㅣ	ㅎ
1 1 1 1 0	Fill Code	Fill Code	Fill Code
1 1 1 1 1	Fill Code	Fill Code	Fill Code

2바이트 조합형 한글은 앞의 세가지 방법중 가장 먼저 개발된 방법으로 어떠한 한글이라도 표현할 수 있다는 장점을 갖고 있다. NKP도 이방식을 소프트웨어로 구현한 것이다. 모든 한글이 가능하므로 워드프로세서(문서작성)에서 이용할 수 있다. 대체로 널리 쓰이고 있는 보석글도 이 방식을 사용하고 있는데 이러한 점은 한글데이터 호환성에서도 유리하다고 할 수 있다.

컴퓨터분야의 대기업(삼보, 금성, 삼성)에서 초기에 만든 한글카드는 대부분 이 방식이다. 그러나 삼보에서 작성한 한글 데이터는 금성이나 삼성에서는 호환이 안된다. 이것은 코드배당이 틀리기 때문이다.

(2) 2바이트 7비트 완성형

2바이트 7비트 완성형은 한글의 모양을 완성된 형태로 보관하였다가 출력시키는 하드웨어적인 방법에 쓰이는데 최상위비트를 사용하지 않는 영문자 두자를 한글 한글자와 대응시켜 한글로 표현하고 있다.

영문자에서는 영문 소문자와 영문 대문자의 조합, 자주 쓰이지 않는 특수코드(_,‘, [,\,], {,|,})들의 조합을 만나면 이에 할당된 한글로 바꾼다.

세운한글카드가 이 방식을 쓰고 있는데 dBASE Ⅲ+를 사용할 때 ‘늣BASE Ⅲ+’로 나오는 이유가 ‘dB’가 한글을 만들 조건을 가지므로 그에 배당된 한글 ‘늣’으로 바뀌는 것이다. (표 2에서 보면 ‘dB’가 10행 3번째 조합형인데 표 3의 10행 3번째 조합을 보면 ‘늣’임을 알 수 있다.)

이것은 대부분의 영문 소프트웨어에서 문자를 체크하는 범위의 문자조합으로 한글을 구성했으므로, 영문 소프트웨어에서 별 수정없이 한글이 가능하다는 장점이 있지만 한글을 만들 수 있는 영문조합이 한정되어 있어, 중요한 한글1350여자 만을 쓴다. 즉 ‘dBASE’와 같이 영문과 한글을 동시에 쓸 때 영문조합을 함부로 한글로 바꾸지 못하기 때문이다. 또 C언어와 같이 특수기호(|, %, [, \,])를 쓰는 언어에서는 특수 기호가 한글로 표현되고 한글속에 특수 코드가 숨어 있으므로 주의해서 써야 한다.

참고로 표 2는 한글을 만드는 본래의 영문코드의 조합으로 한글 스위치를 OFF시킨 상태이고 표 3은 표 2의 데이터를 세운한글카드에서 한글 스위치를 ON 시킨 상태에서 나타난 한글인데 여기서 따옴표를 사용한 6개의

조합은 사용되지 않았고 빈자음과(ㅘ…) 빈모음(ㄱ…)을 빼면 1339자의한글을 나타낼 수 있음을 알 수 있다. 표 4는 표 2의 꼭같은 데이터를 고려시스템(7비트 완성형)에 표현된 한글인데 같은 7비트 완성형이라도 코드가 서로 다름을 알 수 있다. (10행 세번째의 조합 dB-늦-늪) 이것은 1350자를 선택하는 중에 선택한 글자가 틀리기 때문이다.

(3) 2바이트 8비트 완성형

2바이트 8비트 완성형은 7비트 완성형에서 쓰지 않았던 최상위 비트를 사용하므로 그래픽코드와의 조합을 사용하여 한글을 만들 수 있는 조합을 확장한 것이다.

2바이트 8비트 완성형 방식은 위의 두 한글방식에 비해서 늦게 나온 방법으로 한글 2350자와 4888자의 한자를 쓸 수 있다. 이 방식을 행정전산망용 KS 한글이라고도 하는데, 최근 각 업체에서 이 방식의 한글을 채택한 컴퓨터가 선보이기 시작했다. 한글을 2350자를 쓰기 때문에 워드프로세서를 사용하는 데는 별지장이 없고 한자도 쓸 수 있다는 특징이 있다. 그러나 2바이트 8비트 조합형과 마찬가지로 그래픽캐릭터가 한글 또는 한자로 표현되고 영문 소프트웨어의 수정이 필요하다.

이 방식의 한글은 아직 보편화되지 않아 통용되는데 조금 더 시간이 필요할 것으로 보인다.

지금까지 한글카드 방식은 살펴 보았는데 한글카드의 한글세팅(활성화)은 하드웨어적으로 하는 것(한글 스위치의 ON/OFF)과 소프트웨어적으로 해야 하는 것이 있다. 한글입력을 가능하게 하는 한글 key 프로그램과 각 컴퓨터 업체에 따라 다른 프로그램을 제공한다. 또 회사에 따라 한글/영문 변환도 키보드에 한글/영문 변환키가 하나 더 달려 있는 것도 있고 〈Shift〉-〈ALT〉 조합을 사용하는 것도 있다.

〔표 4〕 고려시스템의 한글세트

ㄱ	ㄲ	ㄴ	ㄷ	ㄸ	ㄹ	ㅁ	ㅂ	ㅃ	ㅅ	ㅆ	ㅇ	ㅈ	ㅉ	ㅊ	ㅋ	ㅌ	ㅍ	ㅎ	ㅏ	ㅐ	ㅑ	ㅒ	ㅓ	ㅔ	ㅕ	ㅖ	ㅗ	ㅛ	ㅜ	ㅠ	
ㅡ	ㅣ	가	각	간	갇	갈	갉	감	갑	값	갓	갔	강	갖	갗	같	갚	갛	개	객	갠	갤	갬	갭	갯	갱	갸	걀	거	걱	건
걷	걸	검	겁	것	겅	겆	겉	겊	게	겐	겔	겟	겠	겨	격	겪	견	결	겸	겹	겼	경	곁	계	고	곡	곤	곧	골	곪	
곬	곰	곱	곳	공	곶	과	곽	관	괄	광	괘	괜	괭	괴	괸	굉	교	구	국	군	굳	굴	굵	굶	굼	굽	굿	궁	궈	권	
궐	궤	귀	귄	귈	귐	귓	규	균	귤	그	극	근	글	긁	금	급	긋	긍	기	긴	길	김	깁	깃	깊	까	깍	깎	깐	깔	깜
깝	깡	깥	깨	깰	꺼	꺽	꺾	껌	껍	껑	께	껴	꼈	꼬	꼭	꼰	꼴	꼼	꼽	꼿	꽁	꽂	꽃	꽈	꽉	꽤	꽹	꾀	꾜	꾸	
꾹	꾼	꿀	꿇	꿈	꿔	꿩	꿰	뀌	끄	끈	끊	끌	끓	끔	끗	끝	끼	낀	낄	낌	나	낙	낚	난	날	낡	남	납	낫	났	낭
낮	낯	낱	낳	내	낸	낼	냄	냅	냇	냈	냉	냐	냥	너	넉	넋	넌	널	넓	넘	넝	넣	네	넥	넨	넬	넷	녀	녁	년	념
녔	녕	녘	노	녹	논	놀	놈	놉	놋	농	높	놓	놔	놨	뇌	뇨	뇽	누	눈	눌	눕	눠	뉘	뉴	늄	느	늑	는	늘	늙	늠
능	늦	늪	늬	니	닉	닌	닐	님	닙	닝	다	닥	닦	단	닫	달	닭	닮	닳	담	답	닷	당	닻	닿	대	댁	댄	댈	댐	댓
댕	더	덕	던	덜	덞	덤	덥	덧	덩	덫	덮	데	덴	델	뎃	뎅	도	독	돈	돋	돌	돐	돔	돕	동	돛	돼	됐	되	된	될
됨	됩	두	둑	둔	둘	둠	둡	둥	둬	뒀	뒤	뒷	듀	드	득	든	듣	들	듬	듭	듯	등	디	딕	딘	딛	딜	딤	딥	딧	딩
딪	따	딱	딴	딸	땀	땅	때	땐	땜	땡	떠	떡	떤	떨	떫	떴	떵	떼	뗄	또	똑	똥	뙤	뚜	뚝	뚤	뚫	뚱	뛰	뛴	뜀
뜨	뜩	뜬	뜯	뜰	뜸	뜻	띄	띠	띤	라	락	란	랄	람	랍	랏	랐	랑	랗	래	랙	랜	랠	램	랩	랫	랬	랭	랴	략	량
러	럭	런	럴	럼	럽	럿	렀	렁	렇	레	렉	렌	렐	렘	렛	려	력	련	렬	렴	렵	렷	렸	령	례	로	록	론	롤	롬	롭
롯	롱	뢰	료	룡	루	룩	룬	룰	룸	룹	룻	뤄	류	륙	륜	률	륨	륭	르	륵	른	를	름	릅	릇	릉	릎	리	릭	린	릴
림	립	릿	링	마	막	만	많	맏	말	맑	맘	맙	맛	망	맞	맡	맣	매	맥	맨	맬	맴	맵	맹	맺	머	먹	먼	멀	멈	멋
멍	메	멕	멘	멜	멤	멧	며	면	멸	명	몇	모	목	몫	몬	몰	몸	몹	못	몽	묘	무	묵	묶	문	묻	물	묽	뭄	뭇	뭉
뭏	뭐	뭔	뭘	뭣	뮤	뮴	므	믈	믐	미	믹	민	믿	밀	밈	밉	밍	및	밑	바	박	밖	반	받	발	밝	밟	밤	밥	밧	방
밭	배	백	밴	밸	뱀	뱃	뱅	뱉	버	벅	번	벌	범	법	벗	벙	벚	베	벤	벨	벵	벼	벽	변	별	볍	볏	병	볕	보	복
볶	본	볼	봄	봅	봇	봉	봐	봤	뵈	뵐	부	북	분	불	붉	붐	붓	붕	붙	뷔	뷰	브	븐	블	비	빅	빈	빌	빔	빗	빙
빚	빛	빠	빡	빤	빨	빵	빻	빼	빽	뺀	뺄	뺌	뺏	뺑	뺨	뻐	뻔	뻗	뻘	뻥	뻬	뼈	뽀	뽑	뽕	뿌	뿍	뿐	뿔	뿜	쁘
쁜	쁠	쁨	삐	사	삭	삯	산	살	삶	삼	삽	삿	샀	상	샅	새	색	샌	샐	샘	샛	생	샤	샬	샷	샹	서	석	섞	선	설
섬	섭	섯	섰	성	세	섹	센	셀	셈	셉	셋	셔	션	셨	소	속	손	솔	솜	솟	송	솥	쇄	쇠	쇼	숀	수	숙	순	숟	술
숨	숫	숭	숯	숱	숲	숴	쉐	쉘	쉬	쉰	쉴	쉼	쉽	슈	슘	슛	스	슥	슨	슬	슭	슴	습	슷	승	시	식	신	싣	실	싫
심	십	싯	싱	싶	싸	싹	싼	쌀	쌍	쌓	쌔	쌘	써	썩	썰	썼	썽	쎄	쏘	쏙	쏜	쏠	쐈	쑤	쑥	쓰	쓱	쓴	쓸	씀	씁
씌	씨	씩	씬	씰	씹	씻	씽	아	악	안	앉	않	알	앓	암	압	앗	았	앙	앞	애	액	앤	앨	앰	앵	야	약	얀	얄	얇
얌	양	얕	얗	얘	어	억	언	얹	얻	얼	얽	엄	업	없	엇	었	엉	엊	엌	엎	에	엑	엔	엘	엠	엣	엥	여	역	엮	연
열	염	엽	엾	엿	였	영	옆	옇	예	옛	오	옥	온	올	옮	옳	[illegible]	옴	옵	옷	옹	옻	와	왁	완	왈	왔	왕	왜	외	왼
요	욕	욘	용	우	욱	운	울	움	웁	웃	웅	워	웍	원	월	웠	웨	웬	웰	위	윈	윌	윗	윙	유	육	윤	율	융	으	윽
은	을	읊	음	읍	응	의	이	익	인	일	읽	잃	임	입	잇	있	잉	잊	잎	자	작	잔	잖	잘	잠	잡	잣	잤	장	잦	재
잴	잿	쟁	쟈	저	적	전	절	젊	점	접	젓	정	젖	제	젝	젠	젤	젯	져	젼	졌	조	족	존	졸	좀	좁	종	좇	좋	좌
죄	죠	죤	주	죽	준	줄	줌	줍	줏	중	줘	쥐	쥬	즈	즉	즌	즐	즘	즙	증	지	직	진	질	짊	짐	집	짓	징	짖	
짙	짚	짜	짝	짠	짤	짧	짱	째	쨌	쨍	쩌	쩍	쩐	쩔	쩡	쪼	쪽	쫓	쭈	쭉	쭐	쯔	쯤	찌	찍	찐	찔	찜	찟	찡	차
착	찬	찮	찰	참	찹	찻	찼	창	찾	채	책	챈	챔	챙	챠	처	척	천	철	첨	첩	첫	청	체	첵	첼	쳐	쳤	초	촉	
촌	촐	촛	총	촤	촬	최	추	축	춘	출	춤	춥	충	춰	췌	취	츄	츠	측	츰	층	치	칙	친	칠	칡	침	칩	칫	칭	
카	칵	칸	칼	캄	캉	캐	캔	캘	캠	캡	커	컥	컨	컬	컴	컵	컷	컸	케	켄	켈	켓	켜	켠	켤	켰	코	콕	콘	콜	콤
콧	콩	쾌	쿠	쿡	쿨	쿵	퀴	퀸	큐	크	큰	클	큼	키	킥	킨	킬	킴	킷	킹	타	탁	탄	탈	탐	탑	탓	탕	태	택	
탠	탤	탬	탱	터	턱	턴	털	텀	텁	텅	테	텍	텐	텔	템	토	톡	톤	톨	톰	톱	통	퇴	투	툭	툴	퉁	튀	튜	트	
특	튼	튿	틀	틈	틋	틔	티	틱	틴	틸	팀	팁	팅	파	팍	팎	판	팔	팜	팝	팟	팠	팡	팥	패	팩	팬	팽	퍼	퍽	펀
펄	펌	펐	페	펜	펠	펴	편	펼	평	폐	포	폭	폰	폴	폼	퐁	표	푸	푹	푼	풀	품	풋	풍	퓨	프	픈	플	픔	피	
픽	핀	필	핌	핍	핏	핑	하	학	한	할	함	합	핫	항	해	핵	핸	햄	햇	했	행	햐	향	허	헌	헐	험	헛	헝	헤	
헥	헬	혀	혁	현	혈	혐	협	혔	형	혜	호	혹	혼	홀	홈	홉	홋	홍	홑	화	확	환	활	홧	황	홰	횃	회	획	횡	효
후	훅	훈	훌	훔	훗	훠	훤	훨	훼	휘	휠	휩	휴	흉	흐	흑	흔	흘	흙	흠	흡	흥	흩	희	흰	히	힌	힐	힘	힛	

최근에는 위의 세가지 한글방식을 선택할 수 있는 한글카드(대우, 고려시스템)도 있는데 한글코드변환이 소프트웨어적으로 이루어지거나 DOS상에서 config.sys로 연결해 주어야 하며 때로는 메모리의 낭비를 가져오기도 한다.

맺음말

컴퓨터에서 한글을 쓰는 문제는 초기에서부터 부딪쳤던 문제인데 각 컴퓨터 회사마다 이 문제를 해결하기 위해 노력했고, 그 결과 회사마다 한글코드와 방식이 다르긴 하지만 컴퓨터에서 조금이나마 한글을 쓸 수 있게 되었다(표 5 참조)

〔표 5〕 각 사별 한글 표현방식

한글카드방식	회 사 이 름	비 고
8비트 조합형	삼보, 큐닉스, 삼성, 금성	
KS행정전산망용	삼보, 금성, 옴니테크	
7비트 완성형	세운한글카드	
8비트/KS한글	삼보	모드선택가능
7/8비트/KS한글	대우, 고려시스템	

처음에는 8비트 컴퓨터에서와 같이 N-바이트로 한글을 표현하려 했으나 2바이트로 한글을 표현하는 것이 일반화 되었고 2바이트로 한글을 표현하는 것도 THP나 NKP와 같이 단일 프로그램으로 된 것과 DOS 형태로 된 것이 생기게 되었다. 특히 큐닉스 한글한자DOS, K-한글한자시스템 등은 제한적이긴 하지만 소프트웨어에서 한자까지 처리할 수있게 해 주었다. PC

디스크 드라이브

하드디스크 드라이브 – 40MB 이하 –

PT225
- 공급처 : PTI
  (02) 745-7853 / 4
- 용량 : 21.4MB(포맷시)
- 기록방식 : MFM
- 데이터 전송속도 : 5.0MB / sec
- R / W헤드 수 : 4개
- 액세스 시간 평균 : 35ms
  최대 : 60ms
- 인터페이스 : ST506 / 412
- 소비전력 : 6.5W
- 가격 : 30만원(VAT 별도)

KL-320
- 공급처 : 동양정밀
  (02) 587-2541~8
- 용량 : 21.41MB(포맷시)
- 기록방식 : MFM
- 데이터 전송속도 : 5.0MB / sec
- 액세스 시간 평균 : 40ms
  최대 : 100ms
- 인터페이스 : ST412
- 크기 : 3.5인치
- 가격 : 23만원(VAT 별도)

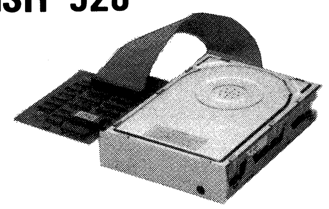
ST 125
- 공급처 : 에이직코리아
  (02) 780-8894
- 용량 : 21.4MB(포맷시)
- 기록방식 : MFM
- 데이터 전송속도 : 5.0MB / sec
- R / W헤드수 : 4개
- 액세스 시간 평균 : 40ms
  최대 : 95ms
- 인터페이스 : ST412
- 크기 : 3.5인치
- 가격 : 28만5천원(VAT 포함)

ST225
- 공급처 : 에이직 코리아
  (02) 780-8894
- 용량 : 21.4MB(포맷시)
- 기록방식 : MFM
- 데이터 전송속도 : 5.0MB / sec
- R / W헤드 수 : 4개
- 액세스 시간 평균 : 65ms
  최대 : 150ms

구입가이드는 애독자 여러분을 위한 란입니다.
본란에는 시스템과 주변기기의 가격정보, 구입시 주의사항 및 자주 발생하는 고장과 이의 응급처치요령 등 PC에 대한 모든 정보를 매월 품목별로 게재하고자 합니다.
게재를 원하는 품목이나 알리고 싶은 제품이 있으시면 본사 편집부 구입가이드 담당자 앞으로 보내주시면 실어드리겠읍니다.
많은 이용이 있으시길 바랍니다.

- 인터페이스 : ST412
- 크기 : 5.25인치
- 가격 : 26만원(VAT 포함)

GSH-520



- 공급처 : 금성통신
  (02) 787-4166~8
- 용량 : 21.3MB(포맷시)
- 기록방식 : MFM
- 데이터 전송속도 : 5.0MB / sec
- R / W헤드 수 : 4개
- 액세스 시간 평균 : 65ms
  최대 : 130ms
- 인터페이스 : SCSI
- 크기 : 5.25인치
- 가격 : 20만원(VAT 별도)

M 8425S
- 공급처 : 창명실업
  (02) 784-5866
- 용량 : 21.3MB(포맷시)
- 기록방식 : RLL(2, 7)
- 데이터 전송속도 : 5.0MB / sec
- R / W 헤드 수 : 4개
- 액세스 시간 평균 : 68ms
  최대 : 150ms
- 인터페이스 : SCSI / 커스텀
- 크기 : 3.5인치
- 가격 : 32만원(VAT 별도)

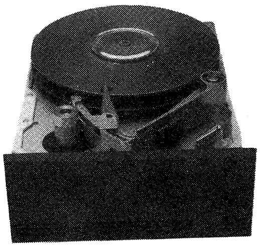
PT238R

- 공급처 : PTI
  (02) 745-7853~4
- 용량 : 32.7MB(포맷시)
- 기록방식 : RLL(2, 7)
- 데이터 전송속도 : 7.5MB / sec
- R / W헤드 수 : 4개
- 액세스 시간 평균 : 35ms
  최대 : 60ms
- 인터페이스 : ST506 / 412
- 소비전력 : 6.5W
- 가격 : 35만원(VAT 별도)

PT338

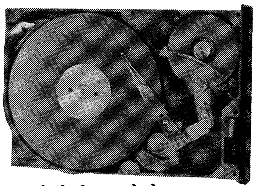
- 공급처 : PTI
  (02) 745-7853~4
- 용량 : 32.0MB(포맷시)
- 기록방식 : MFM
- 데이터 전송속도 : 5.0MB / sec
- R / W헤드 수 : 6개
- 액세스 시간 평균 : 35ms
  최대 : 60ms
- 인터페이스 : ST506 / 412
- 소비전력 : 6.5W
- 가격 : 40만원(VAT 별도)

KL-330



- 공급처 : 동양정밀
  (02) 587-2541~8
- 용량 : 32.73MB(포맷시)
- 기록방식 : RLL
- 데이터 전송속도 : 7.5MB / sec
- 액세스 시간 평균 : 40ms
  최대 : 100ms
- 인터페이스 : ST 412
- 크기 : 3.5인치, 5.25인치
- 가격 : 24만5천원(VAT 별도)

ST138



- 공급처 : 에이직 코리아
  (02) 780-8894
- 용량 : 32.1MB(포맷시)
- 기록방식 : MFM
- 데이터 전송속도 : 5.0MB / sec
- R / W헤드 수 : 6개
- 액세스 시간 평균 : 40ms
  최대 : 95ms
- 인터페이스 : ST412
- 크기 : 3.5인치
- 가격 : 40만원(VAT 포함)

ST238R

- 공급처 : 에이직 코리아
  (02) 780-8894
- 용량 : 32.7MB(포맷시)
- 기록방식 : RLL(2, 7)
- 데이터 전송속도 : 7.5MB / sec
- R / W헤드 수 : 4개
- 액세스 시간 평균 : 65ms
  최대 : 150ms
- 인터페이스 : ST412
- 크기 : 5.25인치
- 가격 : 29만원(VAT 포함)

하드디스크 드라이브 -100MB 이하-

PT351

- 공급처 : PTI
  (02) 745-7853~4
- 용량 : 42.8MB(포맷시)
- 기록방식 : MFM
- 데이터 전송속도 : 5.0MB / sec
- R / W헤드 수 : 6개
- 액세스 시간 평균 : 35ms
  최대 : 60ms
- 인터페이스 : ST506 / 412
- 가격 : 50만원(VAT 별도)

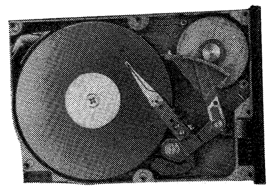
PT357R

- 공급처 : PTI
  (02) 745-7853~4
- 용량 : 49.1MB(포맷시)
- 기록방식 : RLL(2, 7)
- 데이터 전송속도 : 7.5MB / sec
- R / W헤드 수 : 6개
- 액세스 시간 평균 : 35ms
  최대 : 60ms
- 인터페이스 : SCSI
- 가격 : 45만원(VAT 별도)

KL-341

- 공급처 : 동양정밀
  (02) 587-2541
- 용량 : 42.82MB(포맷시)
- 기록방식 : RLL
- 데이터 전송속도 : 10.0MB / sec
- 액세스 시간 평균 : 25ms
  최대 : 75ms
- 인터페이스 : ST412
- 크기 : 3.5인치
- 가격 : 30만원(VAT 별도)

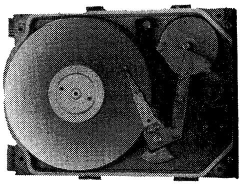
ST157R



- 공급처 : 에이직 코리아
  (02) 780-8894
- 용량 : 49.1MB(포맷시)
- 기록방식 : RLL(2, 7)
- 데이터 전송속도 : 7.5MB / sec
- R / W헤드 수 : 6개
- 액세스 시간 평균 : 28ms
  최대 : 70ms
- 인터페이스 : ST412
- 크기 : 3.5인치

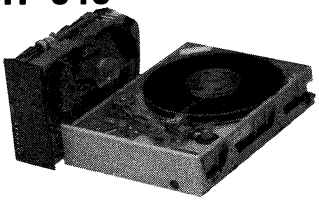
• 가격 : 60만원(VAT 포함)

ST251



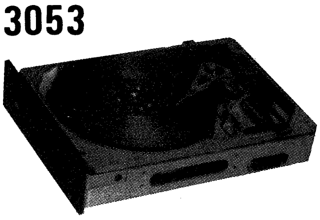
• 공급처 : 에이직 코리아
(02) 780-8894
• 용량 : 42.8MB(포맷시)
• 기록방식 : MFM
• 데이터 전송속도 : 5.0MB / sec
• R / W헤드 수 : 6개
• 액세스 시간 평균 : 40ms
최대 : 95ms
• 인터페이스 : ST412
• 크기 : 5.25인치
• 가격 : 40만원(VAT 포함)

GSH-540



• 공급처 : 금성통신
(02) 787-4166~8
• 용량 : 40.0MB(포맷시)
• 기록방식 : MFM
• 데이터 전송속도 : 6.25MB / sec
• R / W헤드 수 : 4개
• 액세스 시간 평균 : 65ms
최대 : 145ms
• 인터페이스 : SCSI
• 크기 : 5.25인치
• 가격 : 30만원(VAT 별도)

M 3053

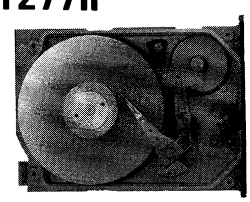


• 공급처 : 창명실업
(02) 784-5866
• 용량 : 44.6MB(포맷시)
• 기록방식 : MFM
• 데이터 전송속도 : 5.0MB / sec
• R / W헤드 수 : 5개
• 액세스 시간 평균 : 25ms
최대 : 55ms
• 인터페이스 : ST412
• 크기 : 5.25인치
• 가격 : 55만원(VAT 별도)

ST4096

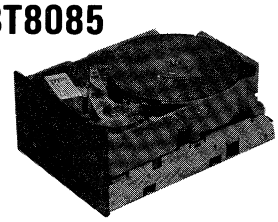
• 공급처 : 에이직 코리아
(02) 780-8894
• 용량 : 80.2MB(포맷시)
• 기록방식 : MFM
• 데이터 전송속도 : 5.0MB / sec
• R / W헤드 수 : 9개
• 액세스 시간 평균 : 28ms
최대 : 55ms
• 인터페이스 : SAT412
• 크기 : 5.25인치
• 가격 : 95만원(VAT 포함)

ST277R



• 공급처 : 에이직 코리아
(02) 780-8894
• 용량 : 65.5MB(포맷시)
• 기록방식 : RLL
• 데이터 전송속도 : 7.5MB / sec
• R / W헤드 수 : 6개
• 액세스 시간 평균 : 40ms
최대 : 95ms
• 인터페이스 : ST412
• 크기 : 5.25인치
• 가격 : 70만원(VAT 포함)

BT8085



• 공급처 : 효성컴퓨터
(02) 545-4151
• 용량 : 71.3MB(포맷시)
• 기록방식 : MFM
• 데이터 전송속도 : 5.0MB / sec
• R / W헤드 수 : 8개
• 액세스 시간 평균 : 25ms
최대 : 55ms
• 인터페이스 : ST506 / 412
• 크기 : 5.25인치
• 가격 : 60만원(VAT 별도)

M 3085



• 공급처 : 창명실업
(02) 784-5866
• 용량 : 71.3MB(포맷시)
• 데이터 전송속도 : 5.0MB / sec
• R / W헤드 수 : 7개
• 액세스 시간 평균 : 22ms
최대 : 50ms
• 인터페이스 : ST412
• 크기 : 5.25인치
• 가격 : 115만원(VAT 별도)

하드디스크 드라이브 -100MB이상-

M 2248E

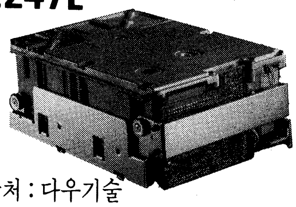
• 공급처 : 다우기술
(02) 556-1251
• 용량 : 224.0MB(포맷시)
• 기록방식 : RLL
• 데이터 전송속도 : 1.5MB / sec

• R / W헤드 수 : 11개
• 액세스 시간 평균 : 18ms
최대 : 35ms
• 인터페이스 : ESDI, SCSI
• 크기 : 5.25인치
• 가격 : 280만원(VAT 별도)

M 2249E

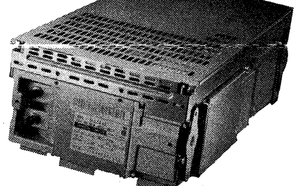
• 공급처 : 다우기술
(02) 556-1251
• 용량 : 305.3MB(포맷시)
• 기록방식 : RLL(1 / 7)
• 데이터 통신속도 : 1.5MB / sec
• 액세스 시간 평균 : 18ms
최대 : 35ms
• 인터페이스 : ESDI, SCSI
• 크기 : 5.25인치
• 가격 : 300만원(VAT 별도)

M 2247E



• 공급처 : 다우기술
(02) 556-1251
• 용량 : 142.5MB(포맷시)
• 기록방식 : RLL(1 / 7)
• 데이터 전송속도 : 1.5MB / sec
• R / W헤드 수 : 7개
• 액세스 시간 평균 : 18ms
최대 : 35ms
• 인터페이스 : ESDI, SCSI
• 크기 : 5.25인치
• 가격 : 260만원(VAT 별도)

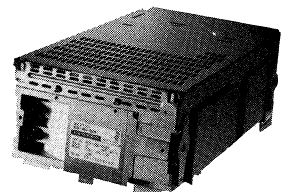
M 2344K



• 공급처 : 다우기술
(02) 556-1251
• 용량 : 690.10MB(언포맷시)

• 기록방식 : RLL(2, 7)
• 데이터 전송속도 : 2.458MB / sec
• R / W헤드 수 : 27개
• 액세스 시간 평균 : 16ms
최대 : 35ms
• 인터페이스 : Modified SMD
• 크기 : 8인치
• 가격 : 500만원(VAT 별도)

M 2372K



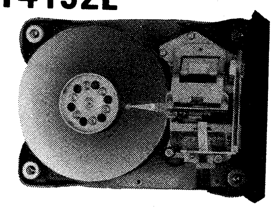
• 공급처 : 다우기술
(02) 556-1251
• 용량 : 823.9MB(언포맷시)
• 기록방식 : RLL(2, 7)
• 데이터 전송속도 : 2.458MB / sec
• R / W헤드 수 : 27개
• 액세스 시간 평균 : 16ms
최대 : 33ms
• 인터페이스 : Modified SMD
• 크기 : 8인치
• 가격 : 530만원(VAT 별도)

ST4144R

• 공급처 : 에이직 코리아
(02) 780-8894
• 용량 : 122.7MB(포맷시)
• 기록방식 : RLL(2, 7)
• 데이터 전송속도 : 7.5MB / sec
• R / W헤드 수 : 9개
• 액세스 시간 평균 : 28ms
최대 : 55ms
• 인터페이스 : ST412
• 크기 : 5.25인치

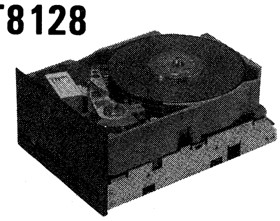
• 가격 : 140만원(VAT 포함)

ST4192E



• 공급처 : 에이직 코리아
(02) 780-8894
• 용량 : 169.1MB(포맷시)
• 기록방식 : RLL(2, 7)
• 데이터 전송속도 : 10MB / sec
• R / W헤드 수 : 8개
• 액세스 시간 평균 : 17ms
최대 : 49ms
• 인터페이스 : ESDI
• 크기 : 5.25인치
• 가격 : 220만원(VAT 포함)

BT8128



• 공급처 : 효성컴퓨터
(02) 545-4151
• 용량 : 110.1MB(포맷시)
• 기록방식 : RLL
• 데이터 전송속도 : 7.5MB / sec
• R / W헤드 수 : 8개
• 액세스 시간 평균 : 25ms
최대 : 55ms
• 인터페이스 : ST506 / 412
• 크기 : 5.25인치
• 가격 : 70만원(VAT 별도)

플로피디스크 드라이브

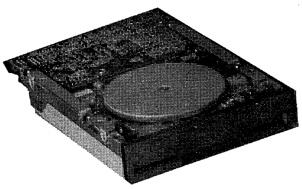
HMF-321

• 공급처 : 현대전자
(02) 741-1311~25
• 용량 : 1.6MB
• 데이터 전송속도 : 500KB / sec
• 기록방식 : FM, MFM



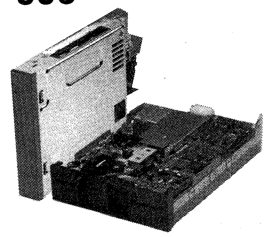
- 회전속도 : 360RPM
- 액세스 시간 평균 : 96ms
- 크기 : 3.5인치
- 가격 : 13만원(VAT 별도)

HMF-311



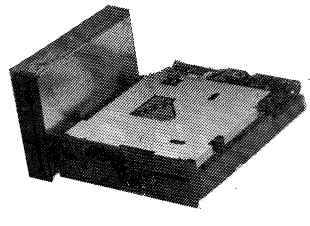
- 공급처 : 현대전자
  (02) 741-1311~25
- 용량 : 1MB
- 데이터 전송속도 : 250KB / sec
- 기록방식 : FM, MFM
- 회전속도 : 300RPM
- 액세스 시간 평균 : 99ms
- 크기 : 3.5인치
- 가격 : 11만원(VAT 별도)

GSF-596



- 공급처 : 금성통신
  (02) 787-4166~8
- 용량 : 1.6MB
- 데이터 전송속도 : 500KB / sec
- 기록방식 : FM, MFM
- 회전속도 : 360RPM
- 액세스 시간 평균 : 94ms
- 크기 : 5.25인치
- 가격 : 7만원(VAT 별도)

GSF-548



- 공급처 : 금성통신
  (02) 787-4166~8
- 용량 : 500KB
- 데이터 전송속도 : 250KB / sec
- 기록방식 : FM, MFM
- 회전속도 : 300PRM
- 액세스 시간 평균 : 93ms
- 크기 : 5.25인치
- 가격 : 5만원(VAT 별도)

보내는 사람

□□□-□□□

우편요금
수취인후납부담
발송유효기간
1989. 2. 1～1991. 12. 31
서울 체신청
승인119호

월간 퍼스널컴퓨터 하이테크

서울특별시 용산구 갈월동 17
퍼스널컴퓨터 편집부앞

1 4 0 - 1 5 0

애독자페이지

관심있는 분야		게재를 원하는 기사	
□ 컴퓨터에 관한 알고 싶은 질문			
□ 본지에 바라는 의견			

정기구독신청서

구독신청방법	□ 은행지로 □ 우체국 대체구좌 □ 소액환 □ 기타		
구독기간	년 월호부터 년 월호까지	구독신청일자	
□ 받으실 분의 이름과 주소(우편번호) ☎ (—)			

※ 이곳에 풀칠을 하여 뒷면 수신란이 나오게 접어서 우체통에 넣어 주세요(우표부착 불필요).

애독자 설문지

책의 편집에 반영하고자 하오니 성의껏 적어서 보내주시기 바랍니다.

작성자	이름		나이		직장		성별	
	주소	☎ (　　—　　)					

사용기종	8비트		16비트		32비트	
	주변기기	□ 모니터(컬러, 단색)　□ 드라이브(HDD, FDD)　□ 프린터(9핀, 24핀, 레이저)　□ 인터페이스카드(　　　)				
	사용용도	□ 취미　□ 업무　□ 교육　□ 학습　□ 기타			사용경력	
	사용언어	□ 베이직　□ C　□ 파스칼　□ 기계어　□ 기타				
	사용분야		사용 소프트웨어			

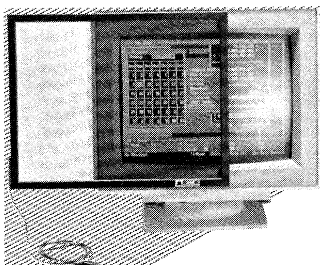
책구입방법	구입방법	□ 정기구독　□ 서점구입(　　　서점)　□ 증정　□ 기타		
	구입동기	□ 정기구독　□ 필요한 내용이 있어서　□ 주위에서 권해서　□ 기타		
	본지외 구독잡지	□ 없다　□ 있다(잡지명 :　　　)	원하는 발행일자	
	애로사항	□ 가격이 비싸다　□ 종이 질이 나쁘다　□ 인쇄상태가 나쁘다　□ 기타		
	□ 본지 구독을 추천하고 싶은 사람이 있다면(이름과 주소) ☎ (　　—　　)			

이번달에 좋았던 기사(2가지)	제목		제목	
	□ 이유		□ 이유	

이번달에 싫었던 기사(2가지)	제목		제목	
	□ 이유		□ 이유	

애독자 선물대잔치

월간 퍼스널컴퓨터는 독자 여러분의 의견을 소중히 하여 책의 편집에 반영하고자 합니다.
애독자 설문지를 사용하여 질의내용에 성의껏 답하시어
15일까지 보내주시면 32명을 뽑아 다음의 선물을 드립니다.



2명

폴리에스터에 특수 구리도금과 아크릴 수지로 가공 처리된 첨단기술의 V.D.T 증후군 방지용 컴퓨터 보안기이다.

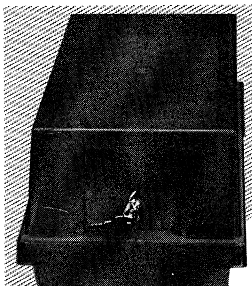
삼종실업 제공
TEL : 784-6931



5명

컴퓨터에 전문지식이 없는 사람도 쉽게 접근할 수 있도록 꾸며진 좋은 지침서이다.

영진출판사 제공
TEL : 719-1119



5명

5.25인치 디스켓을 110장이나 저장할 수 있으며, 자물쇠장치가 부착되어 있어 디스켓을 안전하게 보관할 수 있다.

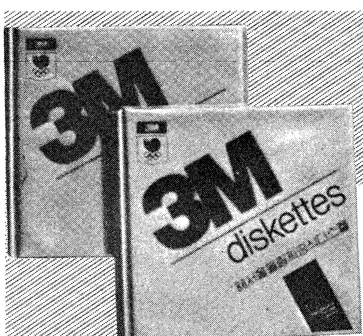
진컴퓨터 제공
TEL : 272-7806



5명

벧엘전자에서는 시스템과 주변기기, 소모품 등을 염가로 판매한다.

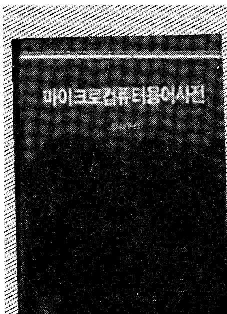
벧엘전자 제공
TEL : 275-2887



10명

특수 산화물 코딩 처리된 에러율 0%의 디스켓이다.

한국쓰리엠 제공
TEL : 785-4567



5명

누구나가 한권씩은 지니고 있어야 하는 컴퓨터 용어사전으로 비교적 자세히 설명되어 있다.

오므사 제공
TEL : 754-1044

※ 애독자 설문지 89년 2월호 당첨자 명단은 180페이지에 있읍니다.

정기구독자에 한하여 프로그램을 제공해 드립니다.

컴퓨터 보급이 늘어남에 따라 이제는 어디에서나 쉽게 접할 수 있을 만큼 대중화 되었으며 컴퓨터란 용어 자체를 자연스럽게 대화속에 끌어들일 수 있게 되었읍니다. 그러나 아직까지는 우리가 활용할수 있는 소프트웨어가 많이 부족하다고 할 수 있읍니다.
저희 퍼스널컴퓨터에서는 이러한 분들을 위하여 이번달부터 공개소프트웨어를 매달 책에 소개하고 정기구독자에 한하여 프로그램을 제공하고자 합니다.

□ 프로그램 제공안내

직접 본사로 오시면 무료로 복사해 드리며 우송을 원하시는 독자의 경우는 디스켓 구입비(1장당 1,500원)와 송료(자켓포함해서 1,000원)를 본지 뒷면에 있는 지로용지를 이용하여 본사 편집부 공개소프트웨어 담당자 앞으로 보내주시면 됩니다.

□ 문의처

서울시 용산구 갈월동 17 ㊀ 140-150
월간 퍼스널컴퓨터 편집부
공개소프트웨어 담당자
전화 : (02) 754—1044

□ 알림

그동안 문의해 오신 많은 분들께 미안한 마음 금할 길이 없읍니다. 정기구독자에 한하여 무료로 증정해 드리는 프로그램 디스켓은 이제 1/3 정도 채워 졌읍니다.
곧 가득차는 대로 본란을 통해 알려드리고 보내드리겠읍니다.

□ 이번달 게재 프로그램

PCUTIL	.EXE	PC매거진을 전개하는 프로그램
MANUAL	.EXE	매뉴얼을 전개하는 프로그램
ATTR	.COM	화일의 속성을 보여주거나 변경한다.
SETUP	.COM	프린터의 제어코드를 언제든지 변환할 수 있는 램상주 유틸리티이다.
SIZE	.COM	화일의 용량을 검사한다.
STATLINE	.COM	화면의 26라인에 Shift 키의 상태를 보여준다.
UNDEL	.COM	지워진 화일을 복구하여 준다.
UNLOCK	.COM	LOCK.COM으로 걸려진 패스워드를 풀어주는 프로그램
VTREE	.COM	서브 디렉토리의 경로를 도면화하여 보여준다.
WAITASEC	.COM	화면의 지나간 부분을 다시 내려 볼 수 있는 램상주 유틸리티
WELCOME		HELF 메세지
WHERE	.COM	찾고자하는 화일의 경로를 모두 보여준다.
XDEL	.COM	화일을 선택적으로 지워주는 프로그램
MANUAL		PC매거진의 매뉴얼
BAC	.COM	하드디스크 또는 플로피디스크의 한 디렉토리 안에 있는 모든 화일을 백업한다.
BOOTSCRN		Boot Screen
BROWSE	.COM	화일의 내용을 화면에 보여준다.
COLOESET	.COM	컬러 색상을 조정한다.
DDIR	.COM	알파벳 순으로 정렬된 화면을 두 줄로 화면에 출력하여 준다.
DISKSCAN	.COM	디스크의 이상유무를 살펴본다.
DOORS	.COM	지워진 화면을 다시 보여주는 램상주 유틸리티
DOS-EDIT	.COM	커서를 마음대로 움직여 도스 명령어를 다시 입력하지 않고 편집할 수 있게 해준다.
FREE	.COM	디스크의 용량을 알아보는 프로그램
KEEPER	.COM	최근에 지나간 10개의 명령어를 다시 사용할 수 있다.
KEY-FAKE	.COM	프로그램 안에서 키를 두드리지 않고 매뉴 선택 등을 가능하게 해준다.
LOCATE	.COM	디스크안에 찾고자 하는 정보가 어느 곳에 있는지를 살펴본다.
LOCK	.COM	화일에 암호를 만들어 다른 사람이 사용하지 못하게 하여 준다.
LPTPORT	.COM	프린트 포트를 변환하여 주는 유틸리티
MONOGRAF	.DRF	LOTUS'S 1-2-3(Version 1A)상에서 모노크롬모니터를 지원하는 그래픽 드라이버
MOVE	.COM	화일들을 선택적으로 복사한다.
NO	.COM	화일의 삭제나 복사의 경우 특정 화일을 제외시킨다.
NPAD	.COM	화면 우측 상단에 작은 윈도우를 만들어 준다.
POPDIR	.COM	PUSHDIR.COM으로 지정된 디렉토리로 돌아간다.
PR	.COM	화일을 프린터로 출력할 경우 그 포맷을 잡아준다.
PRSWAP	.COM	그래픽 프린터가 아닌 프린터를 사용할 경우 IBM-텍스트 그래픽을 변환하여 준다.
PUSHDIR	.COM	현재 있는 경로를 저장한다.
QUICKEYS	.COM	키보드 입력을 좀더 빠르게 조절하여 준다.
RENDIR	.COM	서브 디렉토리의 이름을 사용자가 정의하는 대로 변경한다.

월간 퍼스널컴퓨터의 정기구독자에게는 아래와 같은 특전이 마련되어 있습니다.

—정기구독자는 본사가 보유한 각종 정보 및 자료를 무료로 배부받으실 수 있습니다.
—본지에 게재되는 각종 프로그램을 디스켓에 담아 무료로 보내드립니다.
—본지 및 본지의 자매지(월간 전자과학) 주최로 개최되는 세미나 및 각종 행사에 무료로 참석하실 수 있습니다.
—본지에서 행하는 각종 부대사업에 할인혜택이 주어집니다.
—구독기간중 책 가격이 인상되더라도 종전가격으로 구독이 가능합니다.

● 구독신청은

—시중은행에서 지로용지를 이용하시거나 우체국에서 대체구좌 또는 소액환으로 송금하시면 됩니다.
은행 지로번호 : 3000667
우체국 대체구좌번호 : 010256—31—0526772

● 정기구독료는

— 1 년 구독시 30,000원
— 6개월 구독시 15,000원

● 구독신청시 주의사항은

—구독기간과 주소(우편번호), 전화번호와 받으실 분의 성명을 정확히 기입하시기 바랍니다.
—배달사고나 기타 변동사항은 본지 영업부로 즉시 연락하시기 바랍니다. 신고전의 분실사고는 책임지지 않습니다.

● 정기구독 문의처는

—서울시 용산구 갈월동 17 ㊞ 140-150
월간 퍼스널컴퓨터 영업부
전 화 : (02)754—1044(대표)
FAX : 757—1049

전국총판

●서울●	종로, 중구, 마포, 용산	명문사	774—6314
	중랑, 성동, 동대문	홍익도서	454—1810
	노원, 성북, 도봉	진원사	981—4884
	강남, 강동, 송파	광일사	415—7330
	관악, 동작	관악사	889—4504
	서대문, 은평	형제사	304—9462
	영등포, 구로	영도사	613—5381
	강서, 양천	연우사	865—8239
		진명서점	272—5871
		관음서적	266—7792
●부산●		종합기술서적	462—2431
		영광도서전시관	805—9500
		북부총판	805—9677
		남부총판	756—2727
●대구●		청운서림	23—7335
		한일서적	23—7101
		학원사	22—0336
●인천●		동양서점	73—3384
●광주●		삼복서점	22—0258
●경기●	부천	대경서점	62—3672
	안양, 안산	사랑글방	57—1916
	수원	동아서점	5—6436
	성남	영광서점	2—6532
●강원●	원주	원주종로서적	42—2943
	삼척	영동사	72—6060
	강릉	삼일사	2—2314
	춘천	대진서점	54—5148
●충북●	충주	문학사	2—2814
	청주	화문당	55—0503
●충남●	천안	동방서적공사	63—3314
	대전	동양서점	22—1538
●경북●	포항	경북서점	2—2717
	구미	중앙서점	52—7915
●경남●	진주	동아서점	42—3768
	진해	학애서점	2—2523
	울산	우정서점	43—6003
	마산	기술서적	56—1417
●전북●	전주	홍지서점	6—5311
	군산	동원서점	42—3776
●전남●	목포	문화당	5—3086
	여수	대양서점	62—2111
●제주●		광양서적	53—5077

책을 마감하고

●낯익은 얼굴들과의 생존확인 모임덕에 더욱 분주했던 달. 아직까지 예전 그대로의 모습인데 벌써 엄마가 된 두친구, 아빠가된 한친구, 세월은 그들에게 많은 변화를 갖게 했지만 느낌은 예전 그대로이다. 옛일을 돌아 보면 씁쓸했던 일도 많았는데 그리워지는 것은…, 그리고 정체모를 설레임이 고개를 드는 것은 봄소식을 들은 탓일까? 〈김현〉

●넷!

언제나 「숨비소리」의 노래를 좋아하는 나에게 사랑하는 이가 노래모음집을 보내 주었다. 너무 기뻐 나머지 난 주위를 아랑곳 않고 술 술 돈키호테로 변해간다.

……

옛날 아주 먼 옛날에 ♪~♬ 마음씨 나쁜 마귀할멈이 살고 있었지. 이히히히… 오늘은 누구를 골려줄까. 그러던 어느날 이 마귀할멈은 수정공을 가지고 심술을 부렸네/이 마귀할멈은 이 세상에서 가장 어여쁜 공주를 잡기 위해/공룡을 시켜 공주를 잡게 했네/이 소식을 들은 이웃나라 왕자는 공주를 구하러 마귀성으로 갔네.

"이얏! 쨍! 쨍! 으르렁…"

햇빛 있는데로 가자/꿈이 있는데로 가자.〈榮〉

●지난 겨울은 유난히도 따뜻했읍니다. 학생들은 3월이 되어서야 새로운 한 해를 맞이하는 기쁨을 나눌 때 입니다. 겨우내 컴퓨터와 보냈던 시간들을 못내 서운해하며 한학년씩 오르는 계절이기도 합니다.

"영어공부 최상의 방법은 하루도 거르지 않고 꾸준히 해야 한다"는 말을 우리는 귀가 따가울 정도로 들었읍니다.

컴퓨터에서도 이와 마찬가지라는 생각이 듭니다.

새학기가 되어 다소 시간이 부족할지라도 각자 용도에 따라 사용에 있어 소나기 퍼붓듯이 무리를 한다든지, 아니면 일체 사용치 않고 구석에 둔다든지 하면 영어공부에서 비효율적인 컴퓨터 길들이기를 했다고 할 수 있죠.

틈틈이 바쁜 시간중에도 짬을 내어 컴퓨터와 낯설지 않도록 신경을 써주세요!

새학기를 맞이하는 독자여러분, 그리고 입학·졸업을 맞이한 독자여러분께도 축하인사를 드립니다.

앞날이 훤하시길… 〈용〉

●컴퓨터
당신은 스캔코드로 느끼지만
나는 감성으로 느낀다오
컴퓨터
당신은 CPU로 판단하지만
나는 이성으로 판단한다오
컴퓨터
당신은 모니터로 표현하지만
나는 감정으로 표현하지요
컴퓨터
당신의 모든 것을 좋아하지만
사랑한다 말할 수 없소
당신의 모습에서
미국말은 자취를 감추고
나랏 말씀으로 가득할 때
신명나는 꽹과리소리에 맞추어
키보드를 미친듯이
두드릴 그날을 위해…

〈石心〉

●둘리가 '돌리'가 되었던 참말로 힘들었던 한 달이었읍니다. 건강한 신체로 건전하게 살아갈 수 있다는 것은 행복한 일입니다. 큰 것을 바라지 않고 성실하게 살아가는 것, 그것이야 말로 우리를 기쁘게 하는 좁은 길인가 봅니다. 새봄과 함께 새롭게 출발하는 친구들이 있읍니다. 캠퍼스의 자유로움을 벗어나 진학과 고시, 또는 사회속에서 이제 자기를 향하여 곧게 걸어가는 그들에게 관대한 사회의 모습이 함께 하였으면 좋겠읍니다. 〈둘리〉